Studentenhandleiding Ontwerppracticum

ET2805

E.A. Hendriks, A. Bakker, A. Frehe, P. Groeneveld,
R. Nouta, C. Verhoeven, S. de Graaf, J. Liedorp, A.J. van Genderen

augustus 2009

Inhoudsopgave

1 Inleiding

In het eerste jaar van de studie Elektrotechniek krijgt de student allerlei basisvaardigheden te leren waarvan het hem of haar niet altijd even duidelijk zal zijn wat zij of hij er in de latere beroepspraktijk mee zal kunnen doen. In het ontwerppracticum krijgt de student de gelegenheid om, gebruikmakend van de in het eerste jaar verworven kennis iets 'concreets' te maken. In het ontwerppracticum zal zij/hij ondervinden dat zij/hij ondanks haar/zijn beperkte kennis in staat is, in samenwerking met medestudenten, een complex elektrotechnisch produkt te kunnen specificeren, ontwerpen en testen. Verder zal zij/hij in het practicum geconfronteerd worden met het feit dat een elektrotechnisch ingenieur niet louter individueel bezig is maar dat het kunnen samenwerken in teamverband essentieel is. Hierdoor zal zij/hij een beter beeld gaan krijgen wat er na het afstuderen van haar of hem verwacht gaat worden. De verwachting is dat hierdoor het ontwerppracticum motiverend zal werken om de studie Elektrotechniek tot een goed einde te volbrengen.

Het ontwerppracticum is verdeeld over twee kwartalen in het tweede studiejaar (1e + 3e of 2e + 4e). In het eerste c.q. tweede kwartaal wordt het produkt gespecificeerd en ontworpen. In het derde c.q. vierde kwartaal wordt het produkt getest. Het tussenliggende kwartaal wordt gebruikt om het ontworpen produkt te laten maken.

In hoofdstuk 2 worden de doelstellingen van het ontwerppracticum geformuleerd. Hoofdstuk 3 vermeldt een aantal regelingen en plichten waaraan men moet voldoen. Hoofdstuk 4 geeft een gedetailleerde beschrijving wat er op de verschillende middagen wordt gedaan. Hierbij is aangegeven welke voorbereidingen gevraagd worden, welke werkvormen gebruikt worden. Tevens zijn hier de begeleidingsrollen, de beoordelingscriteria en de beschikbare infrastructuur beschreven. Hoofdstuk 5 beschrijft een aantal aspecten van het werken in teamverband. Hoofdstuk 6 schetst het beeld van het IC-ontwerptrajekt zoals dat gebruikt wordt in het ontwerpppracticum. In hoofdstuk 7 zijn de inwerkopdrachten te vinden en is een voorbeeld gegeven van hoe een inwerkopdracht wordt uitgewerkt, inclusief het gebruik van de ontwerpomgeving. Hoofdstuk 8 bevat de mogelijke groepsopdrachten. Het beschrijft het te ontwerpen produkt, de specificaties en de randvoorwaarden. In de appendices vindt u o.a. een gedetailleerde beschrijving van de sea-of-gates chip met diverse belangrijke parameters en een beschrijving van de beschikbare cellenbibliotheek.

De hoofdstukken 1 t/m 7 dienen gelezen te worden voordat aan het ontwerppracticum wordt begonnen. Hoofdstukken 8 moet bestudeerd zijn voordat aan de groepsopdracht begonnen wordt. De appendices zullen daar waar nodig gelezen moeten worden.

2 Doelstellingen Ontwerppracticum

Het ontwerppracticum draagt ertoe bij dat de student een groepsproces ervaart en leert begrijpen. Het accent ligt op het werken naar een gemeenschappelijk overeengekomen resultaat en het hanteren van de eigen originaliteit ten behoeve van de groepsactiviteit. In het ontwerppracticum spelen zowel inhoudelijke (elektrotechnische) als sociale doelstellingen een rol. De hieronder genoemde doelstellingen zullen deels een 'leren van', deels een 'kennismaken met' karakter hebben.

- *Het leren werken in groepsverband.*
  In de latere beroepspraktijk zal de ingenieur veelal in teamverband aan een projekt werken. Dit houdt in dat de ingenieur moet kunnen samenwerken en communiceren met anderen. Dit vereist niet alleen vakinhoudelijke kennis, maar ook sociale vaardigheden. Dit wil zeggen een bekwaamheid in het omgaan met mensen in een groep ten behoeve van een met en door de leden van een groep te verrichten taak.
  Belangrijke items hierbij zijn o.a. het kunnen luisteren naar elkaar, de eigen ideeën en gedachten onder woorden brengen, de ander stimuleren en niet deprimeren, leiding accepteren, leiding kunnen geven, etc.

- *Het leren werken met randvoorwaarden.*
  Bij elk projekt waarmee de ingenieur later te maken krijgt zullen randvoorwaarden een rol spelen. Dit kunnen inhoudelijke randvoorwaarden zijn (specificaties) alsmede randvoorwaarden t.a.v. beschikbare tijd en geld. In het practicum zal de student met beide aspecten te maken krijgen. Hij/zij zal de (deel)opdracht goed moeten specificeren en binnen een strak tijdschema realiseren.

- *Het leren maken van een eigen ontwerp als onderdeel van het groepsontwerp.*
  Opdrachten zullen in het algemeen te groot zijn om alleen of met z'n tweeën op te lossen. De ingenieur moet in staat zijn om zo'n groter probleem op te delen in deelproblemen en de specificaties hiervan vast te leggen. Vervolgens moet zij/hij in staat zijn om binnen de gestelde mogelijkheden en afspraken een deelprobleem op te lossen.
  De opdracht in het practicum is ook van zo'n omvang dat de student genoodzaakt wordt om het probleem te reduceren tot kleinere deelproblemen en om tot strakke afspraken te komen waaraan een deeloplossing moet voldoen. Bij de uitvoering van een deeltaak zal hij/zij geconfronteerd worden met de consequenties van het niet naleven van de afspraken.

- *Ontwerpgericht denken: analyse als hulpmiddel bij synthese.*
  De student zal in het ontwerppracticum ontwerpgericht moeten denken. Het zal zijn/haar taak niet zijn om een bestaand systeem te ontleden, maar om een nieuw systeem te ontwerpen of eventueel een bestaand systeem aan te passen. Daarbij is de analyse van het ontwerp een onmisbaar gereedschap bij de synthese.
  In het practicum betekent dit dat de student het ontwerp op verschillende niveaus d.m.v. simulatie nauwkeurig zal moeten analyseren.

- *Leren systematisch ontwerpen met moderne computerhulpmiddelen.*
  De studenten leren ontwerpen op een gestructureerde manier. De systematiek komt sterk naar voren in de hiërarchische werkwijze, waarbij op elk niveau van hiërarchie gesimuleerd wordt.

- *Leren analyseren m.b.v. computers op verschillende niveaus.*
  In het practicum zijn voor de verschillende abstractie-niveaus simulatieprogramma's beschikbaar: op logisch niveau, switch-level en transistor niveau. De student zal tijdens het practicum inzicht moeten krijgen op welk niveau welke simulaties zinvol zijn en welke niet.

- *Het belang leren onderkennen van modellen.*
  De student moet het belang leren inzien van het modelleren, dat niet op elk abstractieniveau alle details meegenomen kunnen worden. Een model moet zo eenvoudig mogelijk zijn, maar ook weer niet zo eenvoudig dat effecten die men wil beschrijven niet zichtbaar worden.

- *Kennismaken met het eigen ontwerp voor en na de fabricage.*
  In het practicum worden de studenten geconfronteerd met het resultaat van het eigen ontwerp. Ze leren vaststellen of het produkt geaccepteerd moet worden volgens de vooraf gestelde criteria.

- *Aanleren van meetvaardigheid.*
  De studenten leren omgaan met testprocedures. Ze moeten in staat zijn om een testplan te genereren en uit te voeren.

- *Leren maken en gebruiken van documentatie.*
  De studenten zullen in het practicum documentatie moeten maken waaruit anderen moeten kunnen opmaken wat de eigenschappen van de schakeling zijn. Deze documentatie moet ook zelf gebruikt worden bij het testen van de ontworpen deelschakelingen. Tijdens het practicum moet gebruik gemaakt worden van gedocumenteerde bibliotheekcellen.

3 Regelingen

3.1 Voorkennis en toelatingseisen

Om aan het practicum te mogen deelnemen moet men een voldoende hebben gehaald voor het vak Digitale Systemen en het Propedeutisch Laboratorium. Verder wordt er gebruik gemaakt van kennis opgedaan bij de vakken Elektronische Circuits en Lineaire Elektrische Circuits.

3.2 Inschrijving

Studenten die aan de toelatingseisen voldoen kunnen zich aanmelden via TAS (http://www.tas.tudelft.nl) voor deelname aan het practicum.

3.3 Groepsbijeenkomsten

In het practicum wordt gewerkt in groepen van 10 studenten. Elke groep krijgt één vaste begeleider van de staf en één student-assistent toegewezen. Per week worden drie practicummiddagen of ochtenden gepland. De aanwezigheid op deze ochtenden en middagen is verplicht.
Indien men door ziekte of bijzondere omstandigheid een middag of ochtend moet verzuimen, dan dient men dit van te voren door te geven aan de begeleiders. Bij meer dan 2 middagen afwezigheid zal er, in overleg met de begeleiders, een inhaalprogramma worden vastgesteld, bijvoorbeeld in de vorm van een extra opdracht.

3.4 Voorbereiding

Naast de geroosterde middagen en ochtenden dient men ook tijd uit te trekken voor de voorbereiding, gemiddeld circa 2 uur per middag of ochtend.
Voor het slagen van het practicum is het belangrijk dat men elke middag en ochtend goed voorbereid aan de start verschijnt. Deze voorbereiding kan bestaan uit het lezen van de practicumhandleiding, het bestuderen van manuals of delen van collegedictaten, het uitwerken c.q. voorbereiden van een opdracht, het voorbereiden van een voordracht, etc. De voorbereiding zal door de begeleiders gecontroleerd worden en meegenomen in de eindbeoordeling.

3.5 Verslagen

Over de inwerkopdrachten, de groepsopdracht en later het testen van de schakeling dient een verslag te worden geschreven.
Het verslag over de inwerkopdracht dient kort te zijn en moet een beschrijving geven van de uitwerking, realisatie en simulatie van de ontworpen schakeling. Het verslag dient **twee weken** na beëindiging van de inwerkopdrachten ingeleverd te zijn. De inwerkopdracht wordt in groepjes van 2 gedaan en men mag met z'n tweeën een verslag schrijven.

Het verslag over de groepsopdracht zal veel omvangrijker zijn. Naast een inleiding en een beschrijving van de schakeling als geheel, dient er een uitgebreide beschrijving van elke ontworpen deelschakeling te worden opgenomen. Ieder groepslid dient een herkenbare bijdrage te leveren Dit verslag zal tevens dienen als documentatie voor de te volgen procedures en metingen in de testfase.
Het verdient aanbeveling om al tijdens het practicum een begin te maken met het samenstellen van het verslag over de groepsopdracht. Het verslag moet uiterlijk **vier weken** na beëindiging van de ontwerpfase van het practicum ingeleverd zijn.
Het meetrapport n.a.v. het testen van de schakeling moet een korte samenvatting zijn van de testresultaten van de ontworpen schakeling en een daaraan verbonden conclusie t.a.v. het ontwerp. Dit meetrapport dient **een week** na het testen en meten ingeleverd te zijn.

3.6 Nabespreking

Aan het eind van de ontwerpfase van het practicum is er een groepsbijeenkomst gepland waarin de ervaringen van het werken in de groep besproken zal worden.

4 Opzet en Organisatie

4.1 Inleiding

In dit hoofdstuk vindt u informatie over de infrastructuur, de begeleiding, de beoordeling en een tijdschema (indeling van de dagdelen). In het eerste kwartaal zijn 19.5 dagdelen geprogrammeerd (3 dagdelen per week, te beginnen vanaf de eerste week in het kwartaal). Eventueel kan er nog 1.5 dagdeel worden gebruikt als uitloop. Twee kwartalen later is verder nog een dagdeel gepland voor het testen van het ontwerp.

Het is overigens niet zo dat de groep zich strikt moet houden aan het voorgestelde tijdschema. De eerste dagdelen liggen redelijk vast, maar de overige dagdelen kunnen in overleg anders worden ingedeeld. Opmerking: hoewel in het volgende deel wordt gesproken over middagen kan een dagdeel voor sommige groepen ook uit een ochtend bestaan.

4.2 Indeling middagen

1. KENNISMAKING 1 middag

   Doel: Kennismaken van de groep en begeleiders. Eerste aanzet tot de ontwikkeling van een *wij-gevoel*. Verduidelijking van de doelstellingen van het practicum.
   Kennismaking met de computer-omgeving. Introductie van LINUX en de te gebruiken PC's. Eerste kennismaking met de ontwerpomgeving.

   Vorm: Eerste deel plenair, tweede deel in groepjes van twee.

   Methode: Kennismakingsspel? Kennismaken met de computer.

   Begeleiding: Alle begeleiders (coördinator, student-assistent en eventueel specialist) zijn de gehele middag aanwezig. De coördinator is vooral actief in het eerste deel van de middag, de student-assistent in het tweede deel.

   Voorbereiding: De studenten moeten hoofdstuk 1 t/m 7 van de practiumhandleiding gelezen hebben. Zie verder ook appendix A. Zo nodig voor zichzelf vragen formuleren t.a.v. onduidelijkheden.

2. INWERKEN 2 middagen

   Doel: Verder vertrouwd raken met de ontwerphulpmiddelen. De studenten leren het synthese-trajekt (functie→schema→layout) en de analyse-hulpmiddelen (extractie en simulatie) te gebruiken. Bovendien leren zij omgaan met de ontwerpomgeving op PC's en het gebruik van VHDL.
   Kennismaking met de sea-of-gates chip en fishbone image.

   Vorm: In groepjes van 2 achter een PC.

   Methode: Aan de hand van een eenvoudige voorbeeldschakeling komen alle relevante ontwerpgereedschappen aan bod. Tevens wordt de hiërarchische ontwerpmethode uitgelegd. Aan het begin van elke middag worden zonodig klassikaal (aan de hele groep) nog enkele aspekten toegelicht.

   Opdracht: In de inwerkfase krijgen de studenten een opdracht uit te voeren, die bestaat uit het ontwerpen van een eenvoudige besturing. Ze doen hiermee ervaring op met het maken

van toestandsdiagrammen welk begrip in het college digitale systemen is geïntroduceerd. Ze maken hier kennis met de celbibliotheek en de mogelijkheid van logische synthese m.b.v. een hardware beschrijvings-taal. Hiërarchie komt hier expliciet aan de orde.

Begeleiding: De ontwerpstappen die moeten worden genomen staan in de handleiding, waardoor de studenten veel zelfstandig kunnen doen. Intensieve (specialistische) begeleiding is echter wel noodzakelijk om de studenten niet te lang op een dood spoor te laten zitten. De student-assistent (en eventueel specialist) zijn gedurende deze middagen altijd aanwezig. Van de coördinator wordt verwacht dat hij het indelen in groepjes coördineert en dat hij tijdens deze periode met elk groepje tenminste een keer contact heeft gehad.

Voorbereiding: Voor een vlotte voortgang van deze 'stoomkursus' is het noodzakelijk dat de studenten tevoren de relevante stof (hoofdstuk 6 en 7 van de practicumhandleiding) bestuderen. Daarnaast zullen ze de opdrachten thuis moeten uitwerken.

Beoordeling: Voorbereiding zal beoordeeld worden middels gesprekjes aan het begin van een middag. Verder zal er beoordeeld worden op het uiteindelijke produkt (werkt het). Over de opdracht moet een kort verslag geschreven worden waarin uitwerking, realisatie en simulatie beschreven staan. Richtlijn voor de omvang van de geschreven tekst is 5 A4's.

3. SYSTEEMSPECIFICATIE 3 middagen.

Doel: Leren ontwerpen in een groep. Opdelen van het systeem in een aantal deelsystemen met eenduidige specificaties. Een aantal implementatie-vormen afwegen op grond van theoretische en praktische aspekten. Deze drie middagen zijn erg belangrijk voor het verdere functioneren van de groep. Gedurende deze middagen moet het *wij-gevoel* verder vorm krijgen, ze moeten het gevoel krijgen dat ze gezamenlijk de klus moeten en kunnen klaren. Aan het eind van deze middagen moet het voor iedereen duidelijk zijn dat het zich niet houden aan de gemaakte afspraken fatale consequenties kan hebben voor de schakelingen van de anderen.

Vorm: In de groep van 10 personen, vergaderzaal met bord. Het lijkt zinvol om deze bijeenkomsten deels als een echte vergadering te structureren. Verder zal de groep regelmatig in deelgroepen ideeën uitwerken welke weer plenair besproken worden.

Methode: Aan het begin van de kursus is de opdracht al uitgereikt, zodat de studenten er tijdens het inwerken al over na kunnen denken. De groep moet tot een organieke opdeling van het systeem komen (d.w.z. er moet een blokschema van de tophiërarchie gemaakt worden). De practicumhandleiding bevat in principe de benodigde informatie over de mogelijke bouwstenen van de implementatie. Ontbrekende informatie over de haalbaarheid van een optie moet door de begeleider verschaft worden. Hierna worden voor elk deelsysteem nauwkeurige schriftelijke specificaties geformuleerd (d.w.z. aantal pinnen, de timing, signaalformaat etc.). Deze moeten ook criteria bevatten op grond waarvan beoordeeld kan worden wanneer de schakeling correct werkt. Bovendien bevat deze beschrijving een schatting van de grootte van elk onderdeel. Hiermee kan dan al een voorlopig 'floorplan' gemaakt worden. Als laatste wordt een schatting van de ontwerpinspanning voor elk deelsysteem (tijdplanning) gemaakt. Afhankelijk daarvan wordt de groep opgedeeld in een aantal 'ontwerpgroepjes' van 2-3 man. Elk ontwerpgroepje zal één deelschakeling ontwerpen.

Begeleiding: Het in de juiste banen leiden van dit groepsproces is niet eenvoudig. In principe moeten de studenten zelf alles doen en heeft de begeleiding slechts een adviserende rol. In

de praktijk zal het waarschijnlijk wel nodig zijn dat de vergadering nadrukkelijk gestructureerd wordt (opdrachten voor de volgende middag, etc.). De begeleiding moet hints geven als de groep het spoor kwijt is. In het uiterste geval moet zelfs een groot deel van de uitwerking door de begeleiding gegeven worden. De coördinator zal zich moeten zien als de directeur van een ontwerpbureau. Hij zal er voor moeten zorgen dat de groep uiteindelijk uit elkaar gaat met haalbare, goed gespecificeerde deelsystemen.
Beide begeleiders zijn gedurende elke middag aanwezig. Indien nodig kan externe deskundigheid van een specialist ingeroepen worden.

Voorbereiding: De studenten dienen vooraf hoofdstuk 8 van de practicumhandleiding te bestuderen. Verder zullen ze thuis het systeem of deelsystemen moeten uitwerken.

Beoordeling: Door de begeleiders (individueel). Hierbij spelen de assertiviteit en creativiteit van de studenten een grote rol, maar ook het kunnen en willen meedenken over andere dan de eigen oplossingen.

4. ONTWERP VAN DE HOOFDOPDRACHT 13 middagen.

Doel: Maken van het ontwerp onder gegeven randvoorwaarden.

Vorm: In ontwerpgroepjes van 2-3 man de nauwkeurig omschreven deelschakelingen beschrijven in VHDL en testen op een correcte werking. Samenstellen van het geheel en testen voordat de layout wordt gegenereerd. Genereren van de layout van de totale schakeling.

Methode: Van de deelschakelingen worden per deelschakeling de VHDL-files voor de schakeling gemaakt. Het zal in het algemeen nodig zijn om, gezien de complexiteit ook de deelschakelingen weer onder te verdelen in sub-schakelingen. De beschrijvingen moeten worden getest en daarna moeten de beschrijvingen worden samengesteld tot een geheel en moet dit ook weer worden getest. Wanneer dit goed werkt kan de layout van het geheel worden gemaakt. Belangrijk is de opdracht zodanig te ontwerpen, *dat de schakeling, nadat hij is vervaardigd, en dus alleen aan de in- en uitgangen van de schakeling kan worden gemeten, nog steeds is te testen*. In de eerste middag wordt een taakverdeling en tijdsplanning gemaakt (welke ook door de begeleiders op haalbaarheid getoetst moet worden).

Begeleiding: Aangezien er nu hele wilde schakelingen gemaakt kunnen worden is een goede technische begeleiding van groot belang. Om vruchteloze arbeid zoveel mogelijk te voorkomen moet een werkplan (met blokschema) door de begeleider goedgekeurd worden voordat de deelschakeling geïmplementeerd wordt. Voorts moet de begeleiding technische hulp bieden. Verder zal de begeleiding moeten stimuleren dat er tussen de ontwerpgroepjes kontakt blijft, b.v. door geregeld met een afvaardiging bij elkaar te komen.
Zowel coördinator als student-assistent (eventueel specialist) zijn elke middag aanwezig. De coördinator zorgt ervoor dat er b.v. aan het begin van elke middag een kort werkoverleg is tussen de ontwerpgroepjes zodat afstemming op elkaars werk gegarandeerd blijft.
Daar waar nodig kunnen de ontwerpgroepjes, in overleg met de student-assistent, externe deskundigheid inroepen. Gedurende het ontwerp zullen de ontwerpgoepjes een presentatie houden over de wijze waarop hun deelschakeling functioneert.

Beoordeling: Begeleiders beoordelen wie wat gedaan heeft.

5. NABESPREKING, EVALUATIE 0.5 middag

Doel: Evaluatie van het groepsproces, de samenwerking en het practicum.

Vorm: Groep van 10, vergaderzaaltje.

Methode: Met ontwerpgroep evalueren: wat ging fout en wat ging goed? Het groepsproces wordt besproken, ondervonden problemen en conflicten geëvalueerd en in een groter kader geplaatst. Koppeling van de eigen ervaring naar ontwerpen in groepen in praktijksituaties. Groepsbeoordeling?

Begeleiding: De begeleiders brengen hun kennis en ervaring in en proberen de ervaringen van de studenten te vertalen naar de praktijk.

(Eind)Beoordeling: De ontwerpfase is nu afgesloten. Binnen 4 weken na afloop van het practicum dient het verslag over de groepsopdracht te worden ingeleverd. Eventueel, indien nodig, kunnen er nog individuele gesprekken plaatsvinden tussen begeleiders en studenten. Pas 2 kwartalen later, na het afronden van het testen en meten, en het inleveren van het meetverslag, zal de definitieve beoordeling voor het practicum worden vastgesteld.

6. TESTEN EN METEN HOOFDONTWERP 1 middag (2 kwartalen later)

Doel: leren meten, werking eigen chip vaststellen.

Vorm: Per deelgroep van het ontwerp testen van (een deel van) de ontworpen chip

Methode: Eerste deel van de middag: In groepjes (delen van) het ontwerp testen met een logic analyzer.
Tweede deel van de middag: Het met de gehele groep plaatsen van de schakeling in de benodigde periferie en testen of de totale schakeling werkt volgens de spacificaties.
Meetverslag maken.

Begeleiding: Specialist en eventueel inzet van student-assistenten.

Beoordeling: Op grond van het meetverslag.

4.3 Infrastructuur

Het ontwerppracticum wordt gehouden in het TU gebouw 35 aan de Cornelis Drebbelweg 5, in de zalen 011 en 010. Per groep is een ruimte beschikbaar met een vergadertafel en een zestal PC's die werken onder LINUX. Verder kan gebruik worden gemaakt van een laserprinter. Om te werken op de PC's dient men gebruik te maken van zijn/haar via netID verkregen gebruikersnaam en password. Zie ook appendix A.

4.4 Begeleiding

Elke groep van 10 studenten krijgt twee vaste begeleiders; een coördinator en een student-assistent. Verder is de meeste middagen ook een specialist aanwezig voor aanvullende begeleiding van de groepjes die op dat moment bezig zijn. De begeleidings-taken zijn globaal als volgt verdeeld:

Taken *groepscoördinator* :

- Structuur aanbrengen in de vergaderingen.

- Groepsproces in de gaten houden. Komt iedereen aan bod, heeft iedereen een inbreng, loopt iemand de kantjes eraf, etc.
- Bewaken van de voorgang van het ontwerpproces. Worden er afspraken gemaakt. Houdt iedereen zich aan de afspraken. Is er een tijdschema. Zijn er deadlines gesteld, etc.
- Opdelen van de groep in subgroepen. Zorgen voor wisselende samenstelling van subgroepen.
- Het in de gaten houden van de communicatie tussen de groepen.
- Bemiddelaar/scheidsrechter.
- Controle op juiste afspraken t.a.v. communicatie.
- Controleren van voortgang ontwerpproces.
- Beoordelen.

Taken *student-assistent* :

- Adviseren op inhoudelijk gebied.
- Andere mogelijkheden onder de aandacht brengen.
- Beantwoorden van technische vragen.
- Assistentie bij gebruik PC's en software-tools.
- Trouble shooting.
- Verzamelen van bugs in software en opdracht.
- Doorverwijzing naar specialist.
- Beoordelen.

Taken *specialist* :

- Aanvulling van de taken van de student-assistent.

4.5 Beoordeling

- De beoordeling resulteert niet in een cijfer maar in een voldoende of niet voldoende beoordeling. Studenten die een niet voldoende beoordeling krijgen moeten een extra opdracht uitvoeren waarvan de omvang zal afhangen van de mate van niet voldoende. Dit ter beoordeling van de begeleiders.

- Een belangrijk criterium is het actief meedoen met de groep. Indien men zich niet voldoende inzet kan dit uitsluiting van het ontwerppracticum tot gevolg hebben.
- Mogelijke beoordelingscriteria (niet in volgorde van belangrijkheid):
  - De voorbereiding.
  - Gedrag in de groep (meedenkend, koppig, inbreng in discussies).
  - Zelfwerkzaamheid. Roept de student te vroeg c.q. te laat de hulp van anderen in.
  - Is de student in staat eigen oplossingen te genereren.
  - Blijft de student te lang stilstaan bij details. Is de student in staat beslissingen te nemen, de knoop door te hakken.
  - Houdt de student zich aan de gemaakte afspraken.
  - Is de student in staat om uitleg te geven over zijn of haar eigen resultaten.
  - Kwaliteit van de mondelinge voordracht.
  - Kwaliteit van het schriftelijk verslag.
  - Werkt de gemaakte (deel)schakeling.
  - Aandeel in het groepswerk.
  - Is de student in staat zijn probleem te specificeren.

5 Werken in Teamverband

5.1 Inleiding

Een van de belangrijkste kenmerken van het ontwerppracticum is, dat er wordt gewerkt in groepen, d.w.z. dat de leden van de groep samen een bepaalde, gemeenschappelijke einddoelstelling moeten realiseren.

Werken in groepen betekent, dat de groepsleden niet alleen vak-inhoudelijk bezig zijn met het projekt, maar ook dat zij

- samen moeten kunnen werken (werken met anderen, teamwork) en
- samen moeten kunnen communiceren (o.a. discussiëren en vergaderen).

Beide zaken zijn aanmerkelijk minder gemakkelijk dan men op het eerste gezicht zou veronderstellen.

In een team kunnen werken en kunnen communiceren zijn vormen van sociale vaardigheid. Het zijn extra dimensies van projektonderwijs zoals het ontwerppracticum. Bij individuele studie is iedere student immers afzonderlijk bezig met de leerstof.

Een van de voordelen van projektonderwijs is, dat de student gedwongen wordt, naast het opdoen van zuiver technische kennis (het leren van het ”vak”), zich ook bezig te houden met sociale vaardigheden en dat hij er nu al, tijdens zijn studie, ervaring in kan opdoen.

Ook in zijn latere beroepspraktijk zal de ingenieur immers bijna nooit alleen (als enkeling) technisch handelen; ook dan zal hij met anderen moeten samenwerken en dus ook met anderen moeten communiceren.

Helaas ontbreekt in het OP de tijd voor speciale instructie en training in deze sociale vaardigheden. Maar om te voorkomen dat de groepsleden geheel onvoorbereid aan hun taak zouden moeten beginnen, worden in dit hoofdstuk enkele facetten van de genoemde vaardigheden behandeld. Volledigheid is onmogelijk. Maar bestudering van het hier gegeven minimum aan theoretische achtergrond en toepassing van de daaruit af te leiden praktijk-adviezen is een zeer zinnige zaak.

5.2 De drie dimensies van het werken in een groep

In de inleiding werden ”samen kunnen werken” en ”samen kunnen praten” vormen van sociale vaardigheid genoemd. Wat is sociale vaardigheid? Dat is: bekwaamheid in sociaal handelen, anders gezegd: bekwaamheid in het omgaan met mensen in een groep ten behoeve van een met en door de leden van die groep te verrichten taak. Bij het OP is deze taak de groepsopdracht. Als een groep op weg is naar een doel, dan kunnen aan dit ”op weg zijn” drie aspekten worden onderscheiden:

1. Tijdens de groepsbijeenkomst wisselen de groepsleden, om tot hun gemeenschappelijk doel te komen, verzamelde kennis en informatie uit; ze selekteren de informatie en brengen er een logische samenhang in aan; ze bespreken vaktechnische zaken en resultaten van literatuurstudie en onderzoek; ze rekenen en konstrueren, etc. Kortom: de groepsleden zijn inhoudelijk bezig

met de uitvoering van een taak, met de verwezenlijking van de doelstelling. Men noemt dit de inhoudsdimensie van het groepsgedrag.

2. Aan het bespreken van en de discussie over de inhoudelijke elementen dient - terwille van de begrijpelijkheid, overzichtelijkheid en doelmatigheid - duidelijk vorm te worden gegeven. Er moet volgens bepaalde "spelregels" worden gepraat. Voor de discussie over een probleem moet tijdens de bijeenkomst een verantwoorde werkwijze worden uitgestippeld; de besprekingen dienen gestructureerd te verlopen; men moet methodisch naar oplossingen zoeken, zich beradend op criteria en randvoorwaarden waaraan de aangedragen oplossingen moeten voldoen; besluiten moeten weloverwogen worden genomen. Kortom: de groepsleden moeten tijdens hun besprekingen van de inhoudelijke elementen van het projekt een bepaalde aanpak volgen: ze moeten een procedurele vorm geven aan de inhoud. Men noemt dit de procedure-dimensie van het groepsgedrag.

3. Tijdens de groepsbijeenkomst ontstaat er een bepaalde sfeer: individuen blijven in een groep weliswaar individuen, maar gaan tegelijk ook gedeeltelijk op in het groepsgebeuren. De groepsleden reageren op elkaar, waardoor zij een ander gedragspatroon gaan vertonen dan het louter individuele. Groepsleden beïnvloeden elkaar bewust en onbewust; ze beïnvloeden elkaar positief en negatief, waardoor een klimaat van enthousiasme en coöperativiteit, maar anderzijds ook van vrijblijvendheid of soms zelfs van verveling, irritatie of gedeprimeerdheid kan ontstaan: er ontstaan waarde-oordelen over de personen in de groep en hun bijdragen; er groeit gemotiveerdheid, een wil tot samenwerking en inschikkelijkheid, of anderzijds een neiging tot wedijver en konflikt; er kan sprake zijn van subgroepvorming, met goede of slechte gevolgen voor het groepsgeheel. Kortom: tussen groepsleden groeien onderlinge relaties en gevoelens, vormen van interactie en groepsdynamiek. Men noemt dit de procesdimensie van het groepsgedrag.

Inhoud, procedure en proces zijn dus drie aspekten van het werken in groepen. Deze drie dimensies beïnvloeden elkaar voortdurend. Ze zijn wel te onderscheiden maar niet te scheiden.

Als over inhoudelijke elementen alleen maar chaotisch (ongestructureerd) of in een verziekt klimaat (onwerkbare sfeer) gediscussieerd wordt, komt men niet ver; althans heel wat minder ver dan men zou kunnen komen, en ook met heel wat minder plezier in het werk. Inzicht in en beheersen van de genoemde drie dimensies zijn onontbeerlijk voor het bereiken van een goed eindresultaat in een goede werksfeer.

5.3 De vergadering

De projektvergadering is het moment waarop de groep daadwerkelijk samen aan het werk is. Hier laten mensen zien wat hun vorderingen zijn en hier wordt besloten hoe er verder wordt gegaan. Voor het begin van de vergadering is er een agenda opgesteld die door de voorzitter aan de groep bekend wordt gemaakt (schrijf deze agenda op, of lees de agenda door). Nu weet ieder wat hem/haar te wachten staat gedurende de vergadering. De voorzitter (die de agenda van tevoren opstelt) moet er voor zorgen dat deze een goede opbouw heeft:

- Belangrijke punten eerst: deze mogen nooit door tijdgebrek niet goed aan de orde komen.
- Zorg dat er samenhang is tussen de te behandelen punten (logische opbouw: Als je het een weet, kun je verder met het volgende).

- De agendapunten moeten zo worden geformuleerd dat deelnemers weten waar het precies over gaat.

Een agenda heeft gedeeltelijk een vaste opbouw:

- vaste punten
    - opening en mededelingen
    - wijzigingen in de agenda (uitsluitend als er zeer dringende redenen daarvoor zijn)
    - notulen/verslag + besluitenlijst van de vorige vergadering
    - binnengekomen en verzonden stukken
- variabele punten
    - de eigenlijke agendapunten (in weloverwogen volgorde en goed geformuleerd, zie boven)
- vaste punten
    - vaststelling datum, plaats, tijd en concept-agenda van de volgende vergadering
    - rondvraag
    - sluiting

5.4 Rollen en funkties

Rol van de groepsleden In een vergadering vervult ieder lid van de groep een bepaalde funktie. Wanneer je dus bijvoorbeeld geen voorzitter of notulist bent, ben je toch mede verantwoordelijk voor het goede verloop van de groepsbijeenkomst. Dat betekent in ieder geval een goede voorbereiding van wat er gaat gebeuren (presentatie van het werk dat je gedaan hebt of je gedachten bepalen over een bepaald onderwerp). De volgende personen hebben een bijzondere funktie in de vergadering:

De voorzitter De voorzitter is verantwoordelijk voor een efficiënt verloop van de vergadering. Dat betekent dat er voor hem/haar relatief veel tijd in de voorbereiding gestoken moet worden. Aan de hand van de notulen van de vorige vergadering en m.b.v. het tijd-werkschema kan een agenda opgesteld worden. Van tevoren moet de voorzitter in gedachten bepalen wat het uiteindelijke resultaat moet zijn van ieder te behandelen punt: Een besluit, op de hoogte brengen van andere groepsleden of bijvoorbeeld met elkaar van gedachten wisselen om een mening te vormen over een bepaald onderwerp. Wanneer je je dat niet van tevoren realiseert weet gedurende de vergadering misschien helemaal niemand meer waar de groep nu precies mee bezig is.

Het is zinvol om van tevoren te schatten hoeveel tijd een bepaald onderwerp in beslag gaat nemen. Hierdoor voorkom je dat je een te lange agenda maakt voor de beschikbare tijd waardoor er misschien bepaalde beslissingen te overhaast worden genomen. Belangrijke beslispunten dienen vooraf op schrift geformuleerd te worden. Tijdens de vergadering is de voorzitter formeel de leider. Dat betekent dat hij/zij bepaalt wie er aan het woord is en in mag grijpen wanneer er zaken ter sprake komen die volgens hem/haar niets zinnigs bijdragen aan het doel van het agendapunt. De hele groep

moet echter meewerken en stil zijn wanneer de voorzitter aan iemand anders het woord heeft gegeven. Dat lijkt kinderachtig maar komt het ordelijk (snel en duidelijk) vergaderen ten goede. Het ligt voor de hand dat er een goede samenwerking moet zijn tussen de groep en de voorzitter. Wanneer de voorzitter zich te autoritair opstelt zullen de mensen zich of gepasseerd voelen of ongeïnteresseerd raken door de vervelende sfeer die kan ontstaan. Wanneer de voorzitter echter "te vriendelijk" is, zal de vaart snel uit de vergadering zijn. Er zit geen structuur meer in, omdat niemand die er bewust inbrengt. De voorzitter moet dus wel duidelijk aanwezig zijn. Het ideaal lijkt ook hier weer in het midden te liggen. De zgn. democratische stijl van leidinggeven zorgt ervoor dat de voorzitter voortdurend oplet wat er in de groep leeft en van daaruit ook handelt. De democratisch leider ziet de groep als iets waar voorzichtig mee moet worden omgesprongen maar waar wel duidelijk leiding aan moet worden gegeven. Respekt voor elkaar (goed luisteren!) is een van de belangrijkste aspekten voor het slagen van de groepssamenwerking. De voorzitter moet de aandacht van de groepsleden vragen bij het te behandelen onderwerp. Dat kan hij/zij bijvoorbeeld doen door regelmatig samen te vatten wat er besproken is. Het maken van aantekeningen kan daarbij een grote hulp zijn. Soms is het van belang om zaken op het bord of op papier te zetten. Hierdoor wordt de aandacht van de groep "centraal" gehouden. In feite verloopt het oplossen van een probleem, met bijbehorende besluitvorming, in een vergadering net als in het hele projekt (zie paragraaf 5.6):

Fase 1 Beeldvorming: Doelstellingen, randvoorwaarden, uitgangspunten, probleemstellingen, etc. Hierin bepaal je: "Waar hebben we het precies over en wat moet ons resultaat zijn".

Fase 2 Het genereren van oplossingen: Dit is een wat brainstormachtig gebeuren waarbij alle (voor de groep) denkbare oplossingen op tafel moeten komen. Schrijf alles op!

Fase 3 Concentreer fase: Onder leiding van de voorzitter worden de criteria geformuleerd waarmee de oplossingen beoordeeld gaan worden. In grote lijnen vallen er nu al de nodige oplossingen af. Al pratend over de oplossingen zullen er meer criteria komen en tenslotte komt men bij de besluitvorming.

Fase 4 Besluitvorming: De voorzitter moet hier zorgvuldig te werk gaan. Spreek duidelijk af waarover precies besloten gaat worden en zorg dat niemand "vergeten" wordt. Iedereen moet de kans hebben zijn/haar voorstel te verdedigen om te voorkomen dat men er later op terug wil komen.

Fase 5 Besluit-uitvoering: Hierin moet afgesproken worden wie wat gaat doen. Probeer het werk eerlijk te verdelen. Vriendjes gaan graag samen in een subgroep waardoor er wel eens mensen uit de boot vallen en met een minder boeiend onderwerp worden opgescheept. De voorzitter moet in de volgende vergadering de uitvoering van het besluit kontroleren.

De notulist Van iedere vergadering moet een verslag gemaakt worden. Dat is van belang om onduidelijkheid over de genomen besluiten te voorkomen en kontrole uit te kunnen oefenen. Zorg er dus ook voor dat de notulen bewaard worden en voor iedereen te vinden zijn (b.v. ruim voor de volgende vergadering in het postvak). De volgende zaken moeten in de notulen vermeld staan: Of men kiest voor uitgebreide notulen of voor een beknopt verslag, altijd zullen de volgende zaken vermeld moeten worden:

- datum en tijdsduur van de vergadering

- aanwezig/afwezig (namen van de aanwezigen opsommen - funktie van voorzitter en verslaglegger apart bij de desbetreffende namen vermelden -, bij de afwezigen vermelden of niet met of zonder bericht van verhindering is).
- weergave van de behandeling van alle agendapunten (volgorde van de agenda aanhouden). altijd vermelden:
  - wat is er besloten?
  - werkafspraken (wie doet wat voor wanneer?)
- besluitenlijst (deze is tweeledig):
  - opsomming van alle genomen besluiten en gemaakte afspraken.
  - opsomming van besluiten en werkafspraken uit vroegere vergaderingen, die om welke reden dan ook nog niet zijn afgewerkt.

NB.: In de notulen/het verslag dient objektief te worden vermeld, wat er gezegd/gebeurd is. Geen eigen meningen of kommentaar van de verslaglegger. Ook de eigen deelneming aan de discussie behoort door de verslaglegger objektief te worden weergegeven.

Wanneer de groep gekozen heeft voor een roulerend voorzitterschap (een andere voorzitter per vergadering), kan het nuttig zijn om de notulist van periode ”n” de voorzitter voor periode ”n+1” te laten zijn. Deze weet dan precies wat er gaande is.

De taken van de deelnemers Het projektgroepswerk, met daarin als belangrijke komponenten de discussie en vergadering, is een gezamenlijk proces van alle groepsleden, van voorzitter en deelnemers.
Te vaak denken de deelnemers dat het wel en wee van een bijeenkomst uitsluitend in handen ligt van de voorzitter. Maar zelfs de beste voorzitter moet falen als de deelnemers niet meewerken.
De taak van de voorzitter is veelomvattend en niet gemakkelijk; van de deelnemers mag men vragen, dat zij de voorzitter de uitvoering van zijn taak mogelijk maken. Dat is hun voornaamste taak; alle andere vloeien eigenlijk daaruit voort. Indirekt zijn ze op de vorige bladzijden al ter sprake geweest. We zetten de voornaamste, voor zover zij de groepsvergadering betreffen, nu nog even op een rij.

- Verwacht mag worden, dat alle deelnemers met funkties (verslaglegging. secretariaat, archivering) loyaal met de voorzitter meewerken en in goed overleg steeds tijdig hun plichten vervullen.
- Alle deelnemers behoren zich voor te bereiden op de bijeenkomst. Dat begint al met ”kleinigheden” als op tijd aanwezig zijn. In feite is het opzettelijk of nonchalant te laat komen een belediging jegens de andere deelnemers en de voorzitter, die wel de moeite hebben genomen om op tijd te komen; ”het vergeten” van een vergadering getuigt evenmin van een juiste instelling, nog afgezien van het feit dat zo’n vergadering daardoor kan inboeten aan doelmatigheid. Er ontstaat vaak extra werk in een later stadium. De echte voorbereiding omvat natuurlijk ook het zich bezinnen op de agendapunten: het tevoren bestuderen van documentatie-materiaal; het prepareren van bijdragen, niet alleen qua inhoud, maar ook qua presentatie en formulering, het verzamelen van (eventueel visueel) bewijsmateriaal bij argumenten, etc.

- Alle deelnemers die in een vorige vergadering taken opgedragen gekregen hebben, dienen deze voor de volgende nauwgezet te volbrengen.
- Gedurende de vergadering zelf mag niet alleen van de voorzitter, maar ook van de deelnemers verwacht worden dat zij aantekeningen maken; daardoor kunnen veel overbodige herhalingen en onnodige afdwalingen voorkomen worden.
- Deelnemers moeten zich tijdens de vergadering onthouden van gedrag dat een bijeenkomst nodeloos lang doet duren, het goede klimaat verstoort en irritatie wekt bij de andere aanwezigen. Zij moeten daarom o.a.:
  - goed naar elkaar luisteren
  - elkaar laten uitpraten en niet steeds in de rede vallen
  - meewerken aan het binnen de vastgestelde tijdgrenzen blijven (niet het woord nemen als er niets is aan te vullen op wat anderen al gezegd hebben; mensen die omslachtig vorige sprekers herhalen en er vervolgens aan toevoegen dat ze het er helemaal mee eens zijn, zijn beruchte tijdvreters).
  - bij onverhoopt uitlopen van de vergadering niet er vandoor gaan of met tekenen van ongeduld en bedekte verwensingen "de tijd uitzitten" (dit geldt ook voor de rondvraag).
  - loyaal meewerken (bij democratisch genomen besluiten) aan de besluitvoering, ook al was men aanvankelijk tegen.
- Ieder groepslid heeft tot taak, gedurende de vergadering de voorzitter te steunen door het vervullen van informele leiders-taken, voorzover dat in zijn vermogen ligt en het gewenst is.

5.5 Praktische tips voor de organisatie van de projektgroep

- Duidelijkheid is een eerste vereiste voor een goede organisatie.
- Met heldere afspraken kan een belangrijk deel van de duidelijkheid bereikt worden.
- Wanneer dan iedereen zich aan deze afspraken houdt, is reeds veel gewonnen.
- Voortgangskontrole kan dit bevorderen.
- In het Projektonderwijs hangt veel af van de vergaderingen van de projektgroep.
- Een goede voorbereiding op deze vergaderingen is van veel belang voor het slagen ervan.
- De voorzitter moet deze vergadering primair voorbereiden:
  - agenda samenstellen
  - problemen formuleren
  - stukken tevoren rondzenden.
- Er moeten harde afspraken gemaakt worden: wie doet wanneer wat.
- In de notulen dienen deze afspraken te worden vastgelegd. Uitgebreide notulen zijn meestal niet nodig, zelfs ongewenst want dan worden ze niet meer gelezen. Leg geen hele discussies vast! Hooguit de konklusies.

- Loop bij de start van de volgende vergadering deze afspraken na bij wijze van voortgangskontrole: heeft iedereen zijn taak gedaan? Kan iedereen voldoende vooruit? Zullen de gegevens op tijd aanwezig zijn?
- Probeer taken zo vroeg mogelijk te verdelen zodat men zich daar vast op kan voorbereiden.
- Hanteer, zowel bij het maken van afspraken als bij de voortgangskontrole, het tijdschema als uitgangspunt.

5.6 De procedure voor probleemoplossing en besluitvorming

Voor een correcte probleemoplossing en besluitvorming geldt de volgende procedure in vijf fasen:

1. In de eerste fase, die van de beeldvorming of probleemstelling zorgt de voorzitter ervoor, dat allen een helder beeld krijgen van wat het probleem precies inhoudt. Dit is nl. lang niet altijd voor alle deelnemers onmiddellijk duidelijk. Hierbij kunnen de volgende vragen de revue passeren:
   - hoe en in welke context is het probleem ontstaan? (oorzaken)
   - voor wie is het een probleem?
   - is het een eenmalig of structureel probleem?
   - is alle informatie beschikbaar? Welke gegevens ontbreken nog?
   - is het probleem in eerste instantie correct geformuleerd? Wat ontbreekt?

   De beantwoording van deze vragen brengt vaak nieuwe facetten aan het licht of vereenvoudigt het aanvankelijk verwarde beeld. Aan het einde van de beeldvormingsfase blijkt vaak, als resultaat van de discussie, herformulering van het probleem noodzakelijk; in ieder geval moeten alle deelnemers nu een volledig en identiek beeld hebben van het probleem: onvolledige beeldvorming kan zich wreken tijdens de volgende fasen. Het kan nuttig zijn, het probleem in zijn definitieve formulering op het bord te schrijven.
2. In de tweede fase (creatieve fase of brainstorming) leidt de voorzitter het zoeken van de groep naar mogelijke oplossingen. Hij benadrukt dat de leden zich niet hoeven te beperken tot pasklare oplossingen; iedere suggestie is welkom. Wellicht kunnen ze in een latere fase gekombineerd worden tot een werkelijke oplossing. Creatief en associatief denken is tijdens de brainstorming zeer welkom. De brainstorming kan frappante resultaten opleveren op voorwaarde dat de voorzitter niet toestaat, dat de deelnemers onmiddellijk kommentaar geven op voorstellen van andere deelnemers. Dit is noodzakelijk. Een brainstorming is gedoemd te mislukken als deelnemers zich geremd voelen in hun creatief denkproces door mogelijk schampere reacties. De kritische afweging komt pas in fase 3; de creatieve fase is bedoeld voor het vrij genereren van ideeën, en het elkaar enthousiasmeren in een alles-is-mogelijk- niets-is-gek-klimaat. De voorzitter noteert voor allen duidelijk zichtbaar op het bord alle suggesties en ideeën die geopperd worden. Als niemand meer iets aan de inventarisatie heeft toe te voegen, gaat hij over naar de derde fase.
3. In de derde fase (concentratie fase) begint de voorzitter met het samen met de groep formuleren van de criteria waaraan oplossingen moeten voldoen. Dit zou ook aan het einde van de eerste

fase kunnen gebeuren. De kans is dan echter groot dat de groepsleden door hun vroegtijdige kennis van deze criteria zich tijdens de creatieve fase geremd voelen en niet durven komen met "gewaagde" ideeën. Vervolgens worden de suggesties aan de criteria getoetst en waar mogelijk deeloplossingen met elkaar gekombineerd. Eerst laat men nu die voorstellen vallen, die het minst aan de criteria voldoen. Al discussiërend, kombinerend en schrappend komt men tot een steeds strengere selektie, tot er waarschijnlijk enkele haalbare oplossingen overblijven. Is geen enkele oplossing bruikbaar, dan zal men de criteria ruimer moeten stellen en het laatste deel van de procedure moeten herhalen. De voorzitter ziet erop toe dat ook tijdens deze fase alle groepsleden in de gelegenheid zijn hun visie te geven of hun bijdrage te leveren. Dat betekent niet dat iedereen iets moet zeggen, maar wel dat iedereen iets moet kunnen zeggen. Ook ziet hij toe op logische redeneerwijze en argumentatie: voor- en tegenstanders van bepaalde oplossingen willen nog wel eens met louter emotionele middelen hun standpunt verdedigen of het gelijk aan hun zijde krijgen.

4. In fase vier (besluitvorming) wordt de definitieve oplossing gekozen (het besluit genomen). Zonodig herhaalt de voorzitter de overgebleven oplossingen, schrijft ze op het bord met daarnaast de criteria en konsekwenties. Alle pro's en kontra's van de oplossingen worden tegen elkaar afgewogen (eventueel ook op het bord schrijven, ze zijn dan beter met elkaar te vergelijken). Tenslotte valt het besluit. Het prettigst is het, als het besluit unaniem genomen wordt; dan is iedereen voor dezelfde oplossing en zal zich ook geheel voor de uitvoering van het besluit inzetten. Is unanimiteit niet haalbaar, dan is consensus het meest aangewezen principe. Dat betekent dat niemand zich tegen de keuze van een bepaalde oplossing verzet. Zijn er wel tegenstanders, dan wordt het besluit genomen bij meerderheid van stemmen.

5. Is het besluit eenmaal genomen, dan dient de groep zich nog te bezinnen op de uitvoering ervan (vijfde fase: besluit-uitvoering). Er dienen afspraken gemaakt en regelingen getroffen te worden (wie doet wat voor wanneer?). De voorzitter ziet erop toe dat deze afspraken en regelingen in de notulen en het verslag worden opgenomen.

De loyaliteit eist dat groepsleden die aanvankelijk tegen het genomen besluit waren, zich achter het besluit stellen (mits het uiteraard op correcte wijze genomen is) en meewerken aan de besluit-uitvoering. Wellicht is het voor de voorzitter mogelijk, bij de taakverdeling rekening te houden met de aanvankelijke standpuntbepaling van die groepsleden.

5.7 Evaluatie

De evaluatie beoogt beschrijving en waardering van de wijze waarop gewerkt wordt en, daaruit volgend, - voorzover nodig en mogelijk - verbetering van de activiteiten en het gedrag.

Evalueren is eigenlijk een vorm van feedback geven. We zouden kunnen zeggen dat bij de evaluatie feedback wordt gegeven over alle aspekten van het groepswerk: alle activiteiten worden gewaardeerd, en wel aan de hand van criteria (normen), die impliciet of expliciet worden ontleend aan de doelstellingen van de groep.

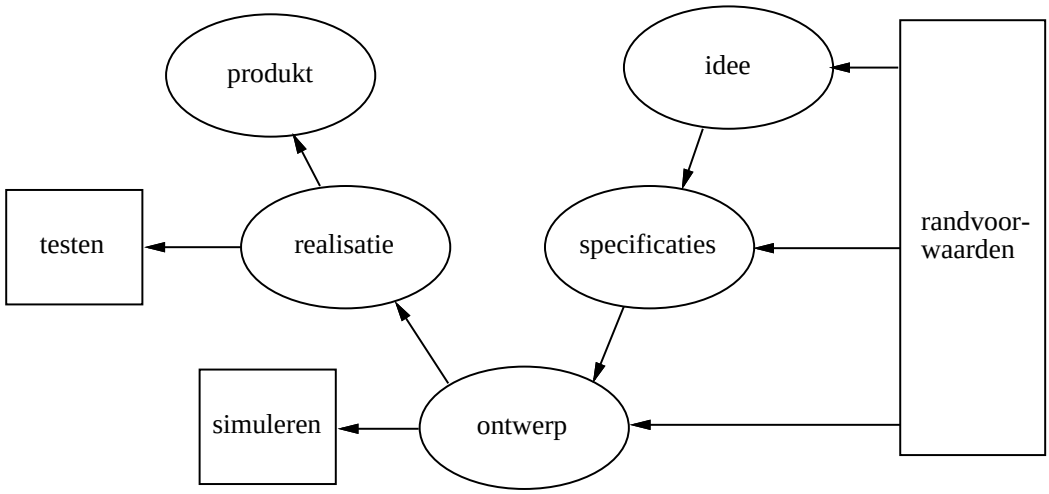
Voorwaarde voor een geslaagde evaluatie is, dat er een open sfeer heerst, waarin ieder groepslid alles ter sprake moet kunnen brengen wat hem op het hart ligt, zonder dat hem dat wordt kwalijk genomen, dus zonder dat andere groepsleden zich persoonlijk gegriefd voelen. Men moet zijn opmerkingen

echter wel kunnen beargumenteren, d.w.z. kunnen zeggen wat men heeft waargenomen (de voor allen zichtbare, objektieve feiten) en hoe men het waargenomene heeft geïnterpreteerd. Waarneming en interpretatie moeten dus steeds duidelijk onderscheiden worden. Tijdens de bespreking blijkt dan wel of de interpretatie van de een overeenkomt met wat de ander bedoelde

6 Het IC Ontwerptrajekt

6.1 Inleiding

Vanaf het moment dat iemand een idee krijgt tot het moment dat het produkt ook echt in de winkel verkrijgbaar is, moeten heel wat ontwerpstappen doorlopen worden. Dit ontwerptrajekt lijkt voor heel veel produkten sterk op elkaar. Zo'n algemeen geldend trajekt is systematisch afgebeeld in figuur 1.



Figuur 1: Algemeen geldend ontwerptrajekt

Ook voor het ontwerp van een ASIC (Application Specific Integrated Circuit) geldt ongeveer dit trajekt. Na het ontstaan van een idee moeten er specificaties opgesteld worden waaraan de schakeling moet voldoen en moeten er ook randvoorwaarden vastgesteld worden. Hierna wordt de schakeling ontworpen. Eerst op circuitniveau en vervolgens de layout. Deze layout wordt geprocessed en vervolgens wordt de ontstane chip getest. Als alles goed is, wordt het sein gegeven om de chip in massaproductie te nemen. Met deze laatste stap zijn we bij het uiteindelijke produkt gekomen.
In het ontwerppracticum doorlopen we dit gehele trajekt vanaf de specificaties tot aan het testen. De nadruk zal echter vooral liggen op het ontwerpen van een IC, dus het verkrijgen van een circuit en layout uit de specificaties en de randvoorwaarden.
Bij het ontwerpen van een IC staan ons computers, software en programmeerbare hardware (voor "rapid prototyping") terzijde. Deze hulpmiddelen zijn in onze tijd onmisbaar vanwege de zeer grote en nog steeds groeiende complexiteit van het hele IC-gebeuren. Ze zijn echter nutteloos als ze niet met verstand worden gebruikt. Bovendien blijft het echte ontwerpen nog steeds het werk van de ingenieur.
Om het ontwerpproces te stroomlijnen is het noodzakelijk enige afspraken te maken met betrekking tot de ontwerpstrategie en het gebruik van de hulpmiddelen. In dit hoofdstuk zullen daarom allereerst enige (ontwerp)regels gegeven worden, waarna vervolgens de tot uw beschikking staande hulpmiddelen aan bod komen.

6.2 Ontwerpregels

Ontwerp hiërarchisch Hiërarchie is absoluut noodzakelijk om een goed overzicht over een ingewikkelde schakeling te kunnen behouden. Als vuistregel kan men stellen dat voor een goed overzicht niet meer dan 5 à 10 deelblokken op één hiërarchisch niveau gedefinieerd moeten worden. Ook de software is gebaseerd op een hiërarchisch ontwerp. Bovendien hebben programma's als de automatische plaatser en bedrader de grootste moeite met grote "plat" ontworpen circuits.
Een uitzondering op deze regel vormen de circuits die met "logische synthese" gemaakt zijn. Hiervoor is het argument van "het overzicht" niet zo belangrijk. De begrenzing ligt dan bij de software. Als vuistregel moet u zich dan houden aan niet meer dan 100 à 200 instances per hiërarchisch niveau.

Verricht eerst een goede verificatie van het volledige circuit m.b.v. simulatie en m.b.v. rapid prototyping, voordat u begint met het maken van de layout In het ontwerptrajekt zijn er talrijke analyse-stappen. Eén van de belangrijkste contrôles die u vaak zult doen, is (computer)simulatie van het ingevoerde circuit. Een andere manier is rapid prototyping, waarbij speciale hardware (een FPGA) zodanig wordt geprogrammeerd dat deze zich gedraagt overeenkomstig de gedragsbeschrijving van het ontwerp. Verricht deze vormen van verificatie voordat u begint met het maken van de layout. Beide manieren vullen elkaar aan en maken het mogelijk fouten in het ontwerp vroegtijdig op te merken en te herstellen. De voordelen zijn kortere iteratie-lussen, wat inhoudt dat men minder ver terug moet in het ontwerptrajekt als iets niet goed blijkt te zijn. Hoewel een grondige verificatie vaak tijdrovend lijkt, blijkt in de praktijk dat deze tijdsinvestering zich ruimschoots terugverdient.

Hou rekening met de hardware bij maken van VHDL-beschrijvingen Hou bij het maken van een VHDL-beschrijving er altijd rekening mee dat de beschrijving een digitaal circuit voorstelt. Hoewel het voordelig is om hoog-niveau VHDL-beschrijvingen te maken (bijvoorbeeld c <= a + b voor de optelling van 2 integer getallen) en deze door de logic synthesizer te laten vertalen naar een netwerk met logische poorten en flipflops, moet altijd in ogenschouw worden genomen dat de VHDL-beschrijving uiteiendelijk wordt afgebeeld op hetzij een combinatorische schakeling, hetzij een sequentiële schakeling volgens het Moore-model. Wanneer de gemaakte VHDL beschrijving niet duidelijk genoeg is m.b.t. het gedrag van de schakeling dan zal de logic synthesizer niet in staat zijn de VHDL gedragsbeschrijving te vertalen naar een structuurbeschrijving, of het gegenereerde circuit zal ongewenst/onvoorspelbaar gedrag vertonen. Meer informatie hierover is te vinden in appendix E.

Ontwerp besturingen volgens het Moore-model Besturingen of sequentiële machines kunnen ontworpen worden volgens het Moore- of het Mealy-model. Het verschil tussen deze twee ontwerpmethoden is dat bij het Moore-model de uitgangssignalen slechts afhankelijk zijn van de toestand van de schakeling, terwijl bij het Mealy-model de uitgangssignalen ook afhankelijk (kunnen) zijn van de ingangssignalen (zie voor meer uitleg appendix D).
In de praktijk heeft dit tot gevolg dat "Moore-besturingen" minder storingsgevoelig zijn en gemakkelijker correct te ontwerpen zijn. Een nadeel is de grotere hoeveelheid toestanden en dientengevolge een groter beslag op het silicium-oppervlak. Dit nadeel weegt echter (zeker tijdens het practicum) niet op tegen de voordelen.

Ontwerp synchroon Synchroon ontwerpen betekent dat alle signalen in een schakeling na een tijdje stopgezet worden om vervolgens op een bepaald tijdstip weer verder te gaan. Dit is noodzakelijk om het overzicht te kunnen houden in de mengelmoes van signalen en hun vertragingstijden die anders zou ontstaan. Het (centrale) signaal dat aangeeft wanneer de schakeling weer verder mag, noemen we de klok. Een ander voordeel van deze synchrone of "geklokte" systemen is dat de problemen die altijd aanwezige spikes kunnen opleveren, zo goed als verwaarloosd kunnen worden: Op het moment dat de flipflops hun nieuwe waarde inlezen, zal hun ingangssignaal altijd stabiel zijn (mits de klokfrequentie laag genoeg gekozen wordt).

Gebruik als geheugenelement alleen edge-triggered flipflops Geheugenelementen kunnen hun nieuwe waarde aannemen tijdens een klokflank (edge) of gedurende de hele periode dat het kloksignaal "hoog" is (pulse). De storingsgevoeligheid van zgn. edge-triggered flipflops is veel lager dan die van zijn pulse-triggered broertje. Om deze reden wordt afgesproken dat tijdens het practicum alleen edge-triggered flipflops zijn toegestaan. De `oplib` (ontwerppracticum-library) bevat trouwens alleen flipflops van dit type.

Ga zorgvuldig om met het kloksignaal Het kloksignaal is het hart van de schakeling. Het is heel belangrijk dat bij alle flipflops de klokflank gelijktijdig aankomt, omdat dat het moment is dat de flipflops inlezen. In de praktijk zal dit echter nooit helemaal lukken. Dit verschijnsel, dat ook wel "clock-skew" wordt genoemd, kan ervoor zorgen dat uw schakeling niet werkt, terwijl hij logisch gezien wel klopt. Ook simulatoren merken clock-skew vaak niet op. Het is dus zaak dat u bij het ontwerpen al met dit verschijnsel rekening houdt en zodanig te werk gaat dat er zo min mogelijk clock-skew ontstaat. Hiervoor zijn in de praktijk een paar regeltjes:

- Gebruik geen logica in uw kloklijn.
- Buffer uw kloksignaal voldoende en gebruik in elke "tak" van uw klok"boom" eenzelfde aantal buffers. Dit betekent dat tussen elke flipflop in uw circuit en het oorspronkelijke kloksignaal hetzelfde aantal buffers staat.
- Belast elke buffer ongeveer even zwaar.

Gebruik geen dynamische (basis)schakelingen Dynamische schakelingen gebruiken de altijd aanwezige capaciteit om tijdelijk informatie op te slaan. Het bekendste voorbeeld van een dynamische schakeling is het dynamische RAM (Random Access Memory), of kortweg DRAM dat tegenwoordig in de meeste computers gebruikt wordt. Het voordeel van dynamische boven gewone "statische" schakelingen is dat ze minder silicium-oppervlak in beslag nemen, wat vooral bij grote repeterende structuren (bijvoorbeeld geheugens) belangrijk is. Nadelen zijn de verplichte aanwezigheid van een meerfase-klok en de slechte testbaarheid bij lage frequenties. Vanwege deze nadelen spreken we in het practicum af dat we geen dynamische geheugencellen of logica zullen gebruiken.

Gebruik alleen positieve logica Signalen kunnen positief en negatief gedefinieerd worden. Voorbeelden van positieve definities zijn "lamp aan" of "persoon gedetecteerd". Een veel gebruikt voorbeeld van een negatieve definitie is "niet-reset" of $\overline{reset}$. Dit laatste wordt vaak gedaan uit oogpunt

van besparing: Het scheelt soms een invertor. Het nadeel van negatieve logica is dat het niet intuïtief is. In gedachten moet steeds weer geïnverteerd worden.
In het practicum spreken we af dat we daarom bij voorkeur positieve logica zullen gebruiken.

Plaats een buffer als een uitgang te zwaar belast wordt Wanneer een uitgang belast wordt, wordt de ”flank” van het uitgaande signaal minder steil. Enerzijds wordt uw schakeling hierdoor trager, anderzijds wordt er naar verhouding meer vermogen gedissipeerd. In uw kloklijn heeft u daarnaast nog het probleem van vergroting van de clock-skew. Om deze effecten binnen de perken te houden, is het belangrijk dat u op tijd buffert. Hiervoor worden bij de bibliotheekcellen de waarden van de ingangscapaciteit en de fanout vermeldt. Als de belasting de fanout-waarde overschrijdt is het verstandig een buffer (b.v. buf40) te plaatsen.

6.3 Hulpmiddelen

6.3.1 IC ontwerpsoftware

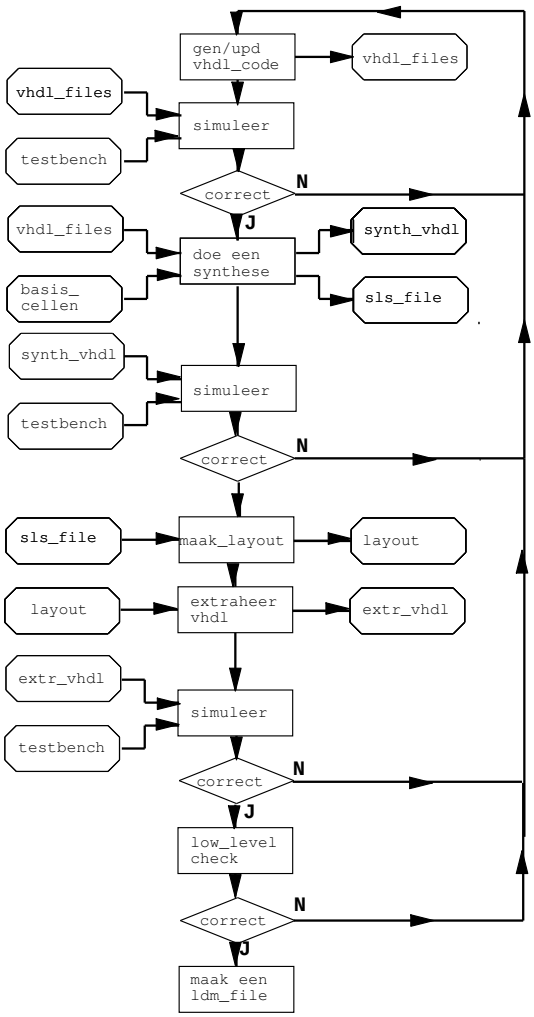
In dit hoofdstuk komen de verschillende computerprogramma’s voor het ontwerpen en controleren van uw schakeling aan bod. Deze programma’s vormen het gereedschap (engels: *tools*), net als een timmerman een hamer en een boormachine gebruikt. Het is niet al te ingewikkeld om met de computergereedschappen om te gaan. U zult er dan ook heel snel mee vertrouwd raken. Het probleem van IC ontwerp is echter dat we met heel veel componenten te maken hebben, die allemaal foutloos op de chip gelegd moeten worden. De computer helpt u hierbij, maar de programma’s zijn vaak niet zo slim en bevatten soms ook fouten. Daardoor kan het voorkomen dat de computer hele rare en onverwachte dingen blijkt te doen. Hiervan moet u niet in paniek raken, want het is heel normaal.

De computerhulpmiddelen die U gaat gebruiken om Uw schakeling te maken, komen van verschillende bronnen. Gedeeltelijk zijn het commercieel verkrijgbare programma’s, zoals het door u in het eerste jaar al gebruikte VHDL-simulatie programma ModelSim, alsmede het synthese programma van de firma Synopsys. Het ontwerpsysteem OCEAN/NELSIS voor het maken van de layouts is een hier op de universiteit zelf door studenten en promovendi ontwikkeld systeem.
In dit hoofdstuk zullen we eerst gaan kijken welke stappen er nodig zijn om een schakeling op Sea-of-Gates te implementeren. Deze stappen zijn aangegeven in figuur 2.

We kunnen hierin de volgende stappen onderscheiden:

- Allereerst moeten uiteraard de VHDL-files worden gemaakt.
  Voor een bepaald onderdeel uit het ontwerp (een ”entity”) kan daarbij een gedrags (”behaviour”) beschrijving of een structurele (”structural”) beschrijving gemaakt worden. Aanwijzingen hiervoor kunnen verder worden gevonden in appendix E. Een speciale file hierbij is verder de file met de VHDL-code voor een testbench. Met behulp van deze testbench kan de schakeling worden getest. De testbench bestaat in principe uit een instantiatie van de schakeling die moet worden getest, met testsignalen die op de ingangen worden aangesloten en uitgangssignalen, waaraan men de correctheid van de schakeling kan aflezen.

- De gemaakte beschrijvingen moeten vervolgens op hun correctheid worden getest m.b.v. een VHDL-simulator. Afhankelijk van de resultaten van deze simulatie kan worden verder gegaan



Figuur 2: De design-flow voor het IC ontwerptraject

met de volgende stap of, bij een niet correcte werking moeten de VHDL-beschrijvingen worden aangepast. De verbeterde beschrijvingen moeten dan weer worden gesimuleerd enz. tot de simulaties het gewenste resultaat opleveren.

- De volgende stap is het synthetiseren van de gemaakte gedragsbeschrijvingen. Hiervoor is, naast de VHDL-beschrijvingen, ook een bibliotheek nodig van basis-circuits waaruit de te ontwerpen schakeling kan worden opgebouwd.
  Bij de synthese kunnen de volgende stappen worden onderscheiden:

  – De VHDL-beschrijvingen worden omgezet in logische formules.

  – Deze formules worden geoptimaliseerd.

  – De geoptimaliseerde formules worden gerealiseerd m.b.v. de cellen uit de bibliotheek.

  - Van de gesynthetiseerde schakeling wordt een structurele VHDL-code gegenereerd, die dus uitsluitend bestaat uit bibliotheek-cellen en hun verbindingen.
    Tevens wordt een SLS-file gemaakt waarin het circuit wordt beschreven. Deze zal worden gebruikt om de layout te maken.

- De gesynthetiseerde schakeling moet hierna weer worden gesimuleerd, Dit kan m.b.v. dezelfde testbench als die ook bij de simulaties van de oorspronkelijke VHDL-code is gebruikt. Alleen moet via een configuratie statement worden aangegeven dat nu de gesynthetiseerde versie van het circuit moet worden gebruikt.
  Wanneer de simulaties uitwijzen, dat het gesynthetiseerde circuit ook de juiste werking vertoont, kan worden verder gegaan met de volgende stap; zo niet, dan moeten de oorspronkelijke VHDL-beschrijvingen zodanig worden aangepast, dat wel het gewenste gesynthetiseerde circuit ontstaat.

- De volgende stap is het vanuit de SLS-beschrijvingen genereren van de layout. Hiertoe kunnen aan de hand van de SLS-beschrijvingen de gebruikte cellen automatisch geplaatst worden door een placement-programma (hiervoor kan het programma *madonna* of de *row placer*. Daarna kunnen de verbindingen tussen de cellen worden verzorgd door het routing-programma *trout*.

- Uit de layout van de schakeling kan weer VHDL-code worden gegenereerd, die nu zal bestaan uit alle basis-cellen van de ontworpen schakeling en hun verbindingen. Deze circuit-extractie gebeurt m.b.v. het programma *space*.

- Ook deze beschrijving moet weer worden gesimuleerd m.b.v. de testbench, waarbij nu, via een configuration statement, moet worden aangegeven dat de geextraheerde versie van de schakeling moet worden getest. Verloopt deze simulatie succesvol, dan kan de volgende stap worden uitgevoerd. Is de simulatie niet succesvol, dan moet uit een nadere analyse van de resultaten van de simulatie worden afgeleid wat er fout is, en wat er aan het ontwerp moet worden veranderd.

- Als voorlaatste stap kan er nog een simulatie worden uitgevoerd op het transistor niveau van de schakeling. Deze stap bestaat uit de volgende sub-stappen:

  - Het transistor-circuit wordt uit de layout geextraheerd en de losse transistoren worden hieruit verwijderd.
  - Vanuit de resultaten van een VHDL-simulatie van het oorspronkelijke circuit worden een command-file en een referentie-file gemaakt. De command-file bevat de ingangssignalen die aan de schakeling moeten worden toegevoegd, en de referentie-file bevat de resultaten van de simulatie.
  - De simulatie wordt nu uitgevoerd m.b.v. de command-file.
  - Het resultaat van de simulatie, een result-file moet nu worden vergeleken met de referentie-file van de simulatie. Komen de twee overeen, dan is het ontwerp klaar en kan worden overgegaan tot de laatste stap. Komen de files niet overeen, dan moet uit een nadere analyse van de verschillen worden afgeleid wat er fout is aan de schakeling en wat hieraan kan worden gedaan.

- Als laatste stap moet er nog een file worden gemaakt waarin de layout van de schakeling staat beschreven (de ldm-file).

6.3.2 Rapid prototyping

Naast de IC ontwerp software zal er ook programmeerbare hardware beschikbaar zijn, in de vorm van een Altera FPGA bord, waarmee het ontwerp ”run-time” geverifieerd kan worden. Deze vorm van verificatie maakt het mogelijk om fouten uit het ontwerp te halen die met simulatie alleen veel moeilijker te achterhalen zijn. Terwijl bij simulatie de gebruiker op logisch niveau alle ingangsignalen moet specificeren en de uitgangsignalen moet interpreteren, zorgen bij rapid prototping de randapparaten ervoor dat de gebruiker op ”hoog niveau” de invoer bepaalt en de uitvoer kan interpreteren. Het is aan te raden verificatie m.b.v. rapid prototyping te verrichten nadat de simulatie van de gedragsbeschrijving van het ontwerp succesvol is afgerond, en voordat met het maken van de layout van het ontwerp wordt begonnen.

De volgende stappen zullen hierbij worden uitgevoerd:

- Na een eerste verificatie met behulp van de VHDL-simulator wordt de VHDL beschrijving van (een gedeelte van) het ontwerp gesynthetiseerd voor de FPGA die aanwezig is op het Altera bord. Dit gebeurt met het programma *quartus*. Gebruik hiervoor alleen behaviour en structural/circuit beschrijvingen, en geen voor de Sea-of-Gates cellen bibliotheek gesynthetiseerde VHDL-beschrijvingen. Op Blackboard staat een template voor een *quartus* project.
- De d.m.v. synthese met *quartus* verkregen configuratie (bitstream) file wordt, eveneens met *quartus*, gebruikt om de FPGA te configureren.
- Op het Altera bord is al aardig wat randapparatuur (schakelaars, LEDs, LCD display, etc.) aanwezig welke gebruikt kan worden tezamen met de FPGA. Eventuele andere randapparaten (luidspreker, monitor, etc.) kunnen ook met het bord worden verbonden.
- Het geprogrammeerde bord wordt ”run-time” op verschillende manieren uitgeprobeerd.
- Wanneer het ontwerp niet aan de eisen voldoet wordt de oorspronkelijke VHDL beschrijving aangepast en wordt weer bij de eerste stap begonnen.

Merk op dat een werkend prototype op het FPGA bord niet automatisch betekend dat de VHDL beschrijving helemaal ”af” is. De ”hardware” van de Sea-of-Gates chip zal zich over het algemeen anders gedragen dan de hardware van het FPGA bord en een goede werking op het bord hoeft geen garantie te zijn voor een goede werking van de chip. Een simulatie van de voor de chip gesynthetiseerde VHDL beschrijving of (nog beter) een simulatie van de layout van de chip, zal hierover meer uitsluitsel geven.

7 Inwerkopdrachten

7.1 Inleiding

Het doel van de inwerkopdracht is om u vertrouwd te maken met de ontwerpomgeving die nodig is om de groepsopdracht te kunnen uitvoeren. Per groep van 2 studenten dient 1 inwerkopdracht gemaakt te worden. De voorbeeld inwerkopdracht over de hotel schakelaar kan doorlopen worden ter voorbereiding van de eigenlijke inwerkopdracht.

Naast de informatie in dit hoofdstuk is verdere uitleg te vinden in het hoofdstuk "Ontwerptrajekt" en de appendices. Het hoofdstuk "Ontwerptrajekt" moet u in ieder geval gelezen hebben voordat u begint aan de inwerkopdrachten.

7.2 Het ontwerpen van een besturing

7.2.1 Inleiding

In deze opdracht wordt aan de hand van een ontwerp van een eenvoudige besturing het hele ontwerptrajekt doorlopen. Allereerst moet de opdracht bestudeerd worden en de specificaties, indien onvolledig, verder vastgesteld worden. Vervolgens moet er op papier een toestandsdiagram gemaakt worden. D.w.z. er moet vastgesteld worden in welke verschillende toestanden de te ontwerpen schakeling kan verkeren en hoe, onder invloed van de stuursignalen, de overgangen plaatsvinden, zie [2]. Uit het toestandsdiagram kan dan een VHDL-beschrijving worden afgeleid en getest, zie [1]. Daarna kan deze beschrijving worden gesynthetiseerd. De gesynthetiseerde schakeling kan worden ingevoerd in het OCEAN/NELSIS systeem en er kan hiermee een layout worden gegenereerd. Na een circuitextractie van de layout kan ook de afgebeelde schakeling op het fishbone image getest worden op zijn functionele werking, zowel op VHDL gate-level niveau als transistor niveau. In appendix D is een uitgebreide beschrijving te vinden van het ontwerpen van besturingen en aan welke voorwaarden zo'n besturing zou moeten voldoen. In appendix E wordt o.a. beschreven hoe zo'n besturing in VHDL kan worden beschreven. Het is raadzaam deze appendices te bestuderen voordat u aan uw eigen ontwerp gaat beginnen.

7.2.2 Het opstellen van een toestandsdiagram

Bestudeer de aan u toegekende opdracht.
Stel een (clock-mode) toestandsdiagram op volgens het Moore-model. D.w.z. dat de uitgangssignalen niet van de ingangssignalen mogen afhangen, maar alleen van de toestand zelf. Ga hiervoor eerst na wat de ingangs- en uitgangssignalen zijn en welke toestanden kunnen optreden. Geef alle toestanden en signalen een functionele naam. Bepaal wat de uitgangssignalen zijn bij elke toestand. Geef de overgangen aan als functie van de stuursignalen. Controleer of alle mogelijke combinaties zijn weergegeven.

7.2.3 Generatie van de besturing

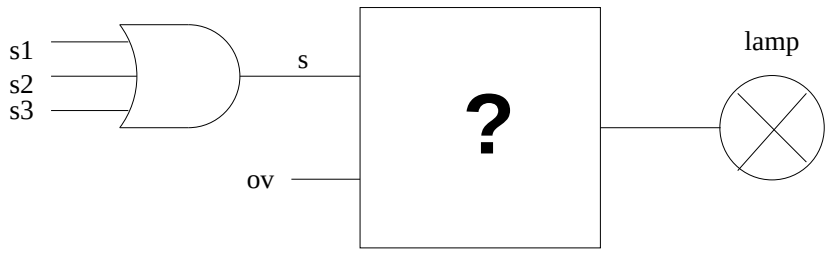
In de volgende paragraaf zal worden besproken welke stappen er moeten worden genomen om uiteindelijk tot een (werkende) schakeling te komen.
Het gehele ontwerp kan worden gedaan m.b.v. de *GoWithTheFlow* interface. Achtereenvolgens dienen nu de volgende stappen hierin te worden doorlopen:

- Maak een VHDL-file, die overeenkomt met de gemaakte FSM, en maak tevens een testbench voor de FSM.
- Compileer de gemaakte VHDL-files
- Simuleer de testbench en kijk of de schakeling zich gedraagt zoals verwacht. Pas zonodig de VHDL-beschrijving aan. Maak tevens de list-file(s) voor latere vergelijking met simulaties van de layout en testen aan de gerealiseerde schakeling.
- Synthetiseer de VHDL-beschrijving van de schakeling. Deze synthese levert twee beschrijvingen:
  - Een nieuwe VHDL-beschrijving, bestaande uit componenten uit de OCEAN/NELSIS-bibliotheek en hun onderlinge verbindingen.
    Deze kunnen worden gecompileerd en dit kan via simulatie worden getest op een correcte werking. Eventueel moet de VHDL-beschrijving worden aangepast.
  - Een beschrijving van het circuit in het SLS-formaat.
    Deze beschrijving kan worden gebruikt als circuit invoer voor het OCEAN/NELSIS systeem.
- Genereer via *madonna* (placer) en *trout* (router) van *seadali* de layout van de schakeling.
- Extraheer de VHDL-code uit de gemaakte layout en simuleer deze om te zien of ook de layout goed functioneert.
- Genereer referentie- en commando-files om de schakeling te kunnen testen op transistor niveau.
- Voer met de gegenereerde commando-file een simulatie uit op het circuit.
- Vergelijk de uitkomst van deze simulatie met de gegenereerde referentie-file. Er mogen geen verschillen optreden.

7.3 Voorbeeld: Een hotelschakelaar

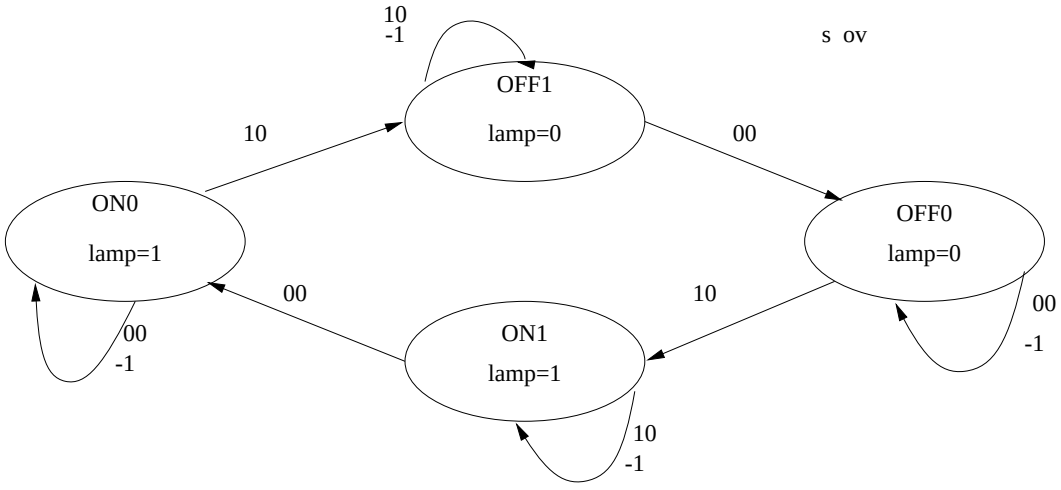
In deze sectie zal een voorbeeld worden gegeven van een inwerkopdracht. We zullen voor het gegeven probleem een toestandsdiagram opstellen, vervolgens de hieruit afgeleide VHDL gedragsbeschrijving invoeren, daarna deze simuleren en synthetiseren, en als laatste de layout maken. Voordat dit voorbeeld wordt doorlopen verdient het aanbeveling de VHDL appendix en de Linux appendix van de Studentenhandleiding Ontwerppracticum te lezen.

Gevraagd: Ontwerp een schakeling die een lamp aanstuurt met 3 toetsen(s1, s2, s3). Wanneer 1 of meer toetsen worden ingedrukt, gaat de lamp aan of uit, afhankelijk van de huidige toestand. Een 'overrule' toets(ov) zorgt dat de lamp in zijn huidige toestand blijft. Maak daartoe de schaking voor de black-box van figuur 3.



Figuur 3: Ontwerpopdracht voor een hotelschakelaar

Uitwerking:
Het toestandsdiagram van figuur 4 kan voor de schakeling worden opgesteld.

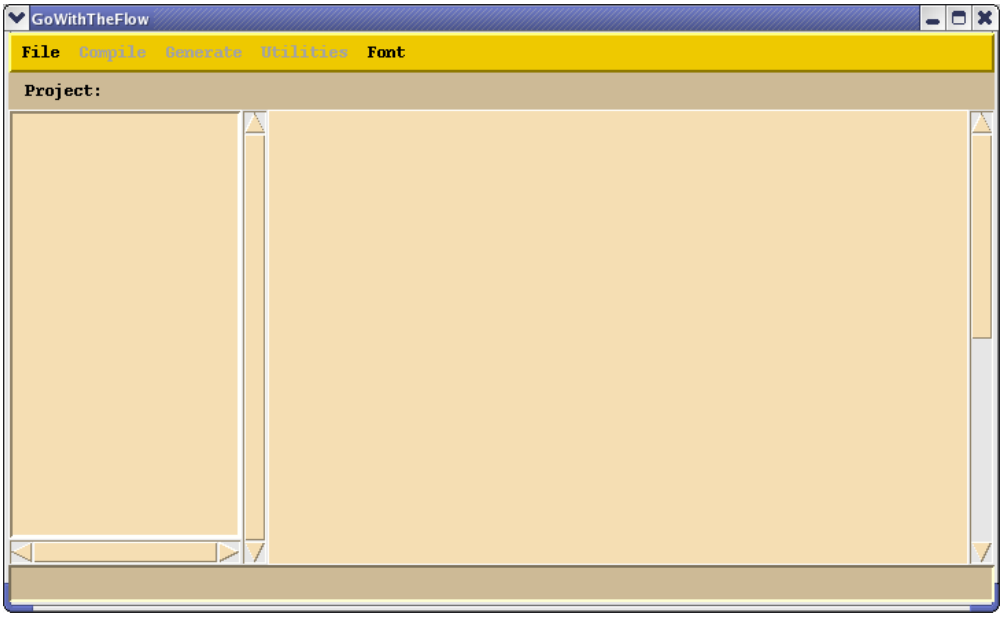


Figuur 4: Toestandsdiagram voor de hotelschakelaar

Zorg eerst dat alle paden naar te gebruiken programmatuur zijn gezet door eenmalig het commando op_init uit te voeren in een terminal window (zie de Linux appendix).

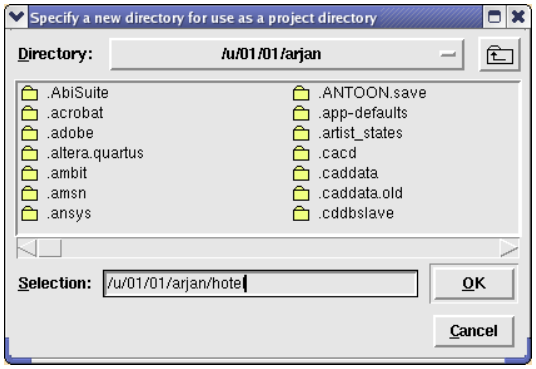
Start nu het programma *GoWithTheFlow* op door op het desbetreffende icon op de DeskTop te

klikken. Dit zal de interface zoals in figuur 5 te zien geven.



Figuur 5: Het programma *GoWithTheFlow* na opstarten

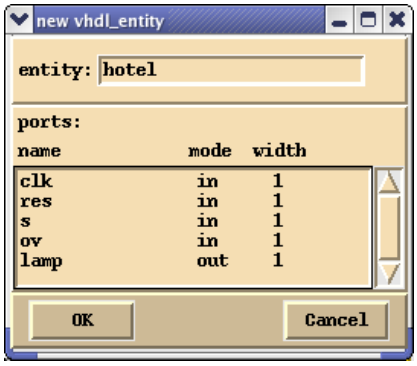
Maak dan een project directory aan m.b.v. het commando File → New project. Een window zoals in figuur 6 zal verschijnen. Specificeer in het vak achter Selection de naam van een nog niet bestaande directory, en klik op OK.



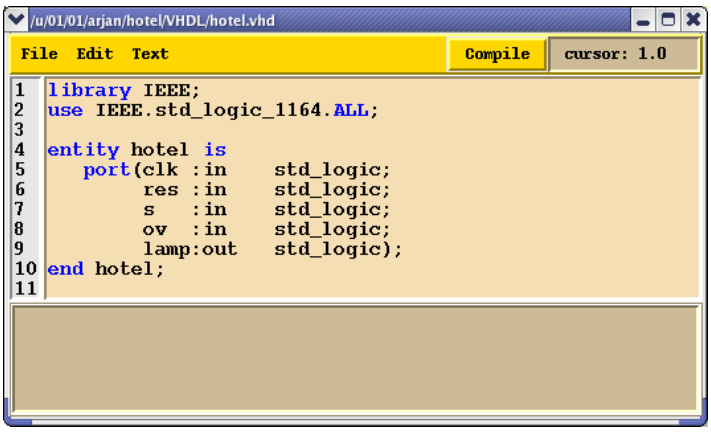
Figuur 6: Het aanmaken van een project directory

Voer dan allereerst de entity beschrijving voor hotel in door op File → New entity te klikken en het daarop verschijnende widget in te vullen zoals in figuur 7. Gebruik de tab toets om naar de volgende kolom te springen.

Na op OK geklikt te hebben verschijnt de bijbehorende VHDL beschrijving in een new window, zie figuur 8. Klik op Compile. Na eerst bevestigend te hebben geantwoord op de vraag of de beschrijving moet worden weggeschreven zal deze worden opgeborgen in de database van *GoWithTheFlow* en vervolgens worden gecompileerd.



Figuur 7: Het invoeren van een nieuwe entity



Figuur 8: De VHDL code voor de hotel entity

Klik vervolgens op de rechthoek ”hotel” en selecteer Add Architecture om de volgende gedragsbeschrijving in te voeren. Merk op dat (conform wat in VHDL appendix is beschreven) het eerste process statement (lbl1) een beschrijving bevat van de toestandsregisters, en het tweede process statement (lbl2) een beschrijving van de combinatorische logica die aan de hand van de huidige toestand en de ingangssignalen de volgende toestand en de uitganssignalen berekent.

```
library IEEE;
use IEEE.std_logic_1164.ALL;

architecture behaviour of hotel is
   type lamp_state is (OFF0, OFF1, ON0, ON1);
   signal state, new_state: lamp_state;
begin
   lbl1: process (clk)
   begin
      if (clk'event and clk = '1') then
         if res = '1' then
            state <= OFF0;
         else
```

```
            state <= new_state;
         end if;
      end if;
   end process;
   lbl2: process(state, s, ov)
   begin
      case state is
         when OFF0 =>
            lamp <= '0';
            if (s = '1') and (ov = '0') then
                new_state <= ON1;
            else
                new_state <= OFF0;
            end if;
         when ON1 =>
            lamp <= '1';
            if (s = '0') and (ov = '0') then
                new_state <= ON0;
            else
                new_state <= ON1;
            end if;
         when ON0 =>
            lamp <= '1';
            if (s = '1') and (ov = '0') then
                new_state <= OFF1;
            else
                new_state <= ON0;
            end if;
         when OFF1 =>
            lamp <= '0';
            if (s = '0') and (ov = '0') then
                new_state <= OFF0;
            else
                new_state <= OFF1;
            end if;
      end case;
   end process;
end behaviour;
```

Klik op Compile. Het programma zal vragen om de beschrijving eerst weg te schrijven. Antwoord bevestigend, en wanneer er geen fouten in de beschrijving zitten zal vervolgens een rechthoek "behaviour" onder het blokje hotel verschijnen ten teken dat de beschrijving is opgeborgen in de database van *GoWithTheFlow*. Wanneer er wel compilatie fouten zijn, zullen deze fouten getoond worden en zullen deze eerst verbeterd moeten worden.

Ga vervolgens op de rechthoek behaviour staan, klik met de linker muisknop en selecteer Add Configuration. Er zal nu een window verschijnen met een naam voor de VHDL configuratie file. Klik

op OK, de VHDL beschrijving voor de configuratie file wordt getoond, schrijf die vervolgens weg en compileer die.

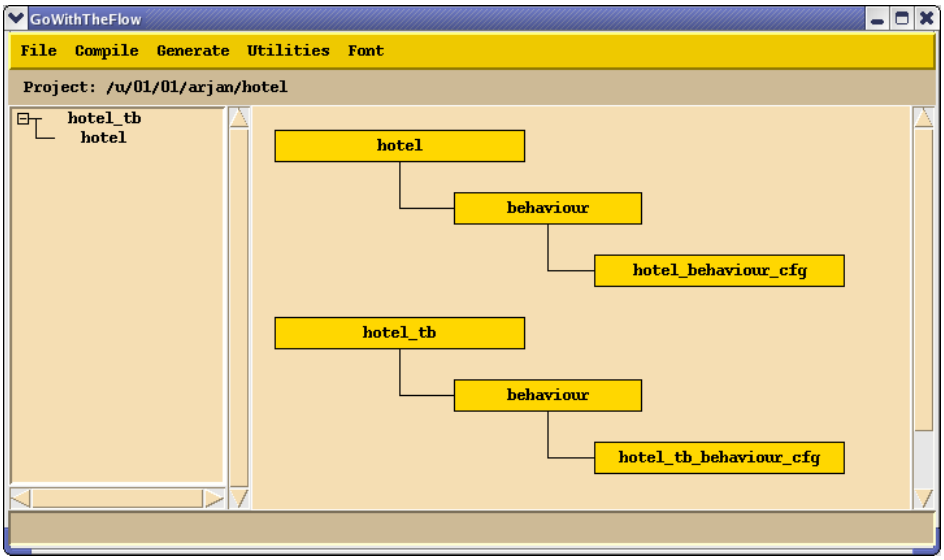
Maak nu op dezelfde wijze ook een entity beschrijving, een gedragsbeschrijving en een configuratie file aan voor een testbench voor de hotel schakeling. Noem de entity "hotel_tb" (het is niet nodig om terminals te specificeren voor deze testbench entity) en maak een behaviour beschrijving als volgt:

```
library IEEE;
use IEEE.std_logic_1164.ALL;

architecture behaviour of hotel_tb is
   component hotel
      port (clk  : in  std_logic;
            res  : in  std_logic;
            s    : in  std_logic;
            ov   : in  std_logic;
            lamp : out std_logic);
   end component;
   signal clk: std_logic;
   signal res: std_logic;
   signal s: std_logic;
   signal ov: std_logic;
   signal lamp: std_logic;
begin
   lbl1: hotel port map (clk, res, s, ov, lamp);
   clk <= '1' after 0 ns,
          '0' after 100 ns when clk /= '0' else '1' after 100 ns;
   res <= '1' after 0 ns,
          '0' after 200 ns;
   s <= '0' after 0 ns,
        '1' after 600 ns,
        '0' after 1000 ns,
        '1' after 1400 ns,
        '0' after 1800 ns,
        '1' after 2200 ns,
        '0' after 2600 ns,
        '1' after 3000 ns,
        '0' after 3400 ns,
        '1' after 3800 ns;
   ov <= '0' after 0 ns,
         '1' after 1800 ns,
         '0' after 2600 ns;
end behaviour;
```

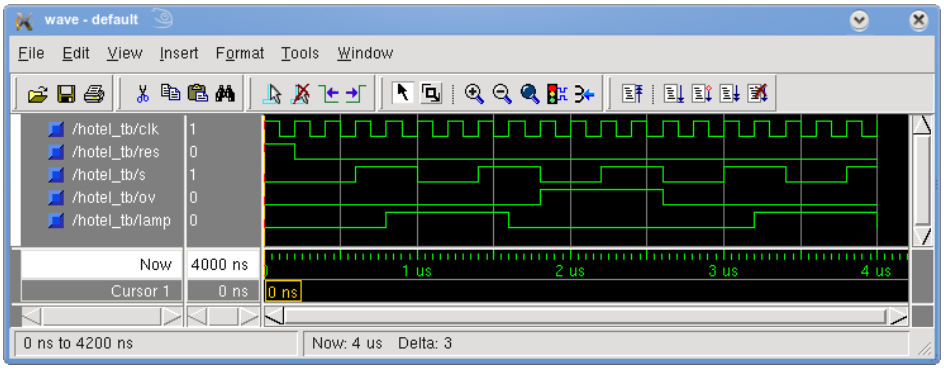
Het main window van *GoWithTheFlow* zou nu een beeld moeten geven zoals in figuur 9.

Start nu een VHDL simulatie door met de linker muisknop op hotel_tb_behaviour_cfg te klikken en



Figuur 9: *GoWithTheFlow* na toevoegen behaviour and testbench

simulate te selecteren. Na een simulatie over bijvoorbeeld 4000 ns zal het wave window van ModelSim een resultaat geven zoals in figuur 10. Om zo dadelijk ook een switch-level simulatie (een



Figuur 10: Resultaat van een of VHDL simulatie met ModelSim

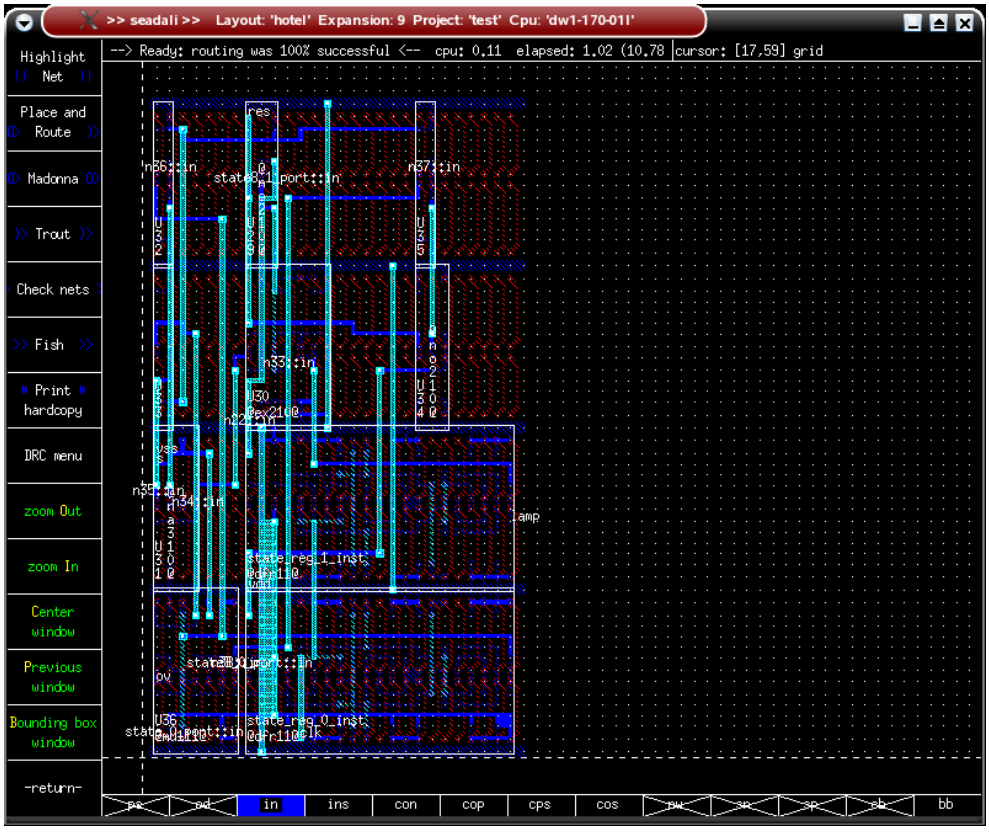
simulatie op transistor niveau) voor de gemaakte layout te kunnen verrichten zullen we nu een .lst file aanmaken waarin informatie over de in en uitgangssignalen van de simulatie staat. Selecteer daartoe in het Tools menu van het wave window het commando Make_list_file. Selecteer vervolgens de hotel behaviour beschrijving en klik op OK. Schrijf vervolgens de file hotel.lst weg.

Vervolgens zullen we een synthese stap doen voor de behaviour beschrijving van de hotel entity. Daarbij wordt de gedragsbeschrijving omgezet in een netwerkbeschrijving bestaande uit cellen uit de Sea-of-Gates celbibliotheek. Klik daartoe met de linker muisknop op de configuratie rechthoek van hotel en selecteer synthesize. Het synthesizer window zal nu verschijnen. Klik makeSyntScript en daarna Synthesize. Eventueel kan op Show circuit worden geklikt om een schema van het gesynthetiseerde circuit te zien. Klik vervolgens op Compile om de VHDL beschrijving te compileren en Parse_sls om een SLS circuit beschrijving in the backend (OCEAN/NELSIS) library te plaatsen. Er zullen nu rechthoeken voor de gesynthetiseerde VHDL beschrijving en de gesynthetiseerde circuit

beschrijving zichtbaar worden in *GoWithTheFlow*.

De gesynthetiseerde VHDL beschrijving kan nu ook gesimuleerd worden met de eerder aangemaakte testbench. Klik daarvoor op de behaviour beschrijving van de testbench en voeg een nieuwe configuratie toe. Geef de nieuwe configuratie een naam die verwijst naar de synthese, bijvoorbeeld hotel_tb_behaviour_syn_cfg. Er zal nu gevraagd worden welke versie van hotel gekozen moet worden voor de nieuwe configuratie van hotel_tb. Selecteer de gesynthetiseerde versie en klik OK. Voor de nieuwe configuratie kan nu een simulatie worden opgestart.

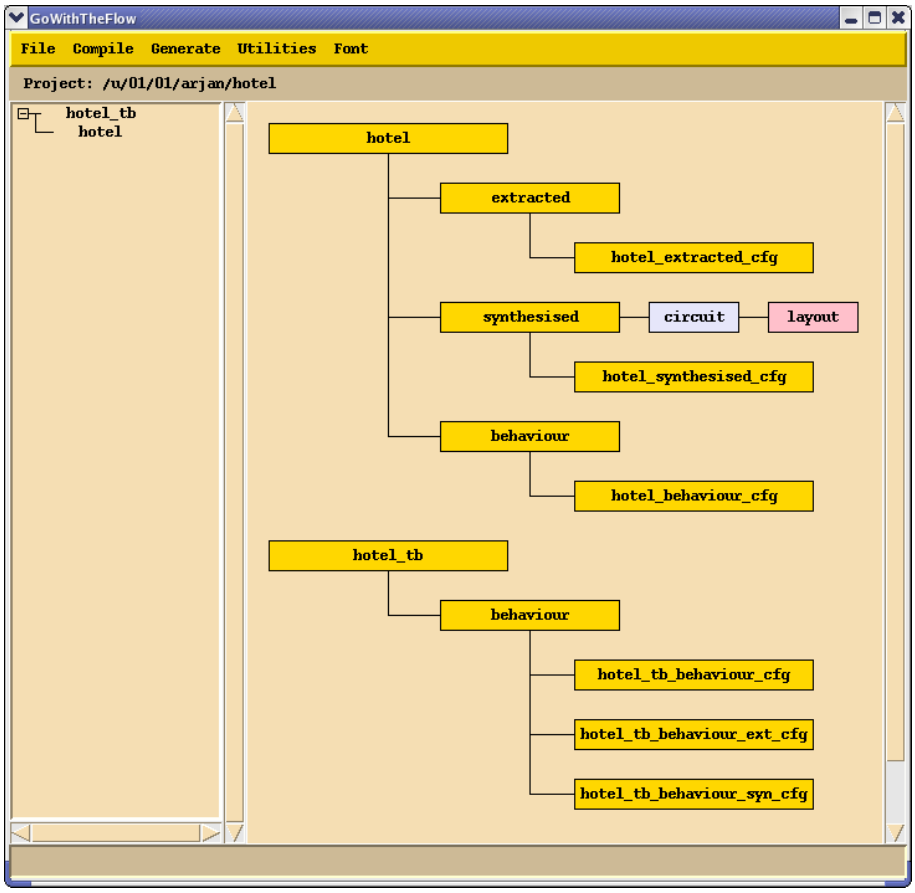
Vanuit de gesynthetiseerde circuit beschrijving kan verder een layout gemaakt worden. Klik daartoe op de rechthoek circuit en selecteer Place & route. Het programma *seadali* zal nu gestart worden. Ga naar het menu automatic tools en voer achtereenvolgens de stappen plaatsen (m.b.v. *madonna*) en bedraden (m.b.v. *trout*) uit. Na afloop zal een layout zoals in figuur 11 te zien zijn. Schrijf vervolgens de layout weg via het database menu (ga terug naar het hoofdmenu via -return-) en verlaat de layout editor.



Figuur 11: Een layout voor de hotel schakeling

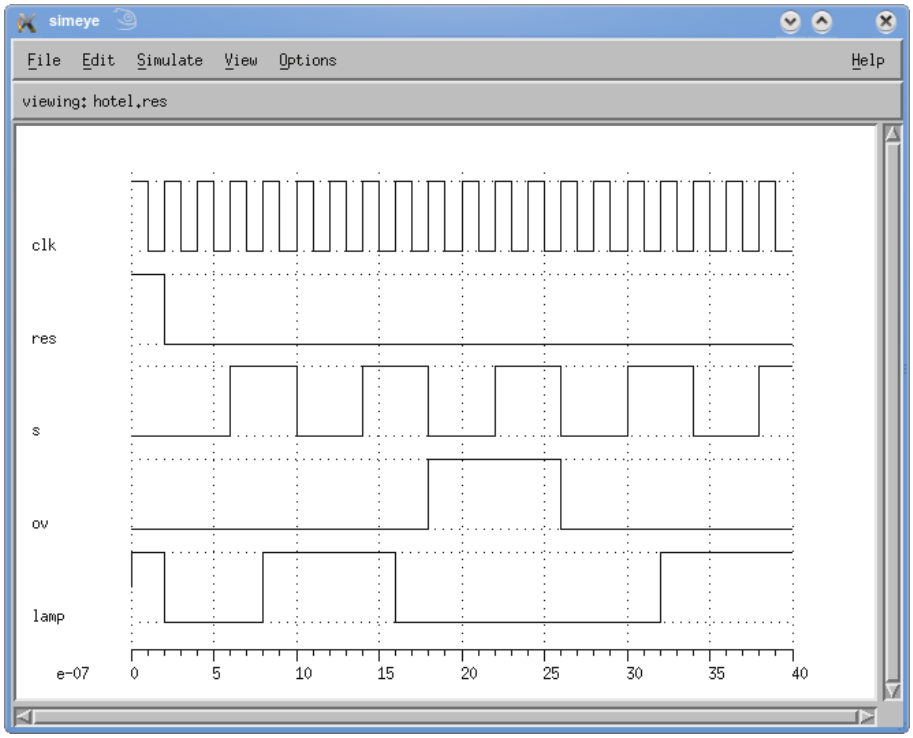
Een alternatief voor Place & route m.b.v. *seadali* bestaat uit het eerst plaatsen van de componenten m.b.v. de *row placer*. Selecteer daartoe op de rechthoek circuit de optie Run row placer. M.b.v. deze placer kan over het algemeen een betere (compactere) plaatsing verkregen worden dan m.b.v. *seadali*. Deze placer kan echter alleen overweg met componenten uit de cellen bibliotheek (en dus niet met hierarchische ontwerpen waarbij eerder door u ontworpen layouts als component worden gebruikt) Verder zal de bedrading altijd nog met *trout* in *seadali* moeten worden gedaan.

De layout kan nu op 2 manieren gecontroleerd worden. De eerste manier bestaat uit het extraheren van een VHDL beschrijving uit de layout en deze met ModelSim te simuleren. Klik daartoe met de linker muisknop op de layout rechthoek en selecteer Extract vhdl. In het nieuwe window, klik Get, vervolgens Write en daarna Compile. Voeg vervolgens een configuratie toe op de manier zoals al eerder gedaan. Maak nu voor de behaviour beschrijving van de testbench voor hotel opnieuw een nieuwe configuratie waarbij de geextraheerde versie van hotel geselecteerd wordt en op deze manier kan de geextraheerde VHDL beschrijving gesimuleerd worden. Het main window van *GoWithTheFlow* zal nu een beeld zoals in figuur 12.



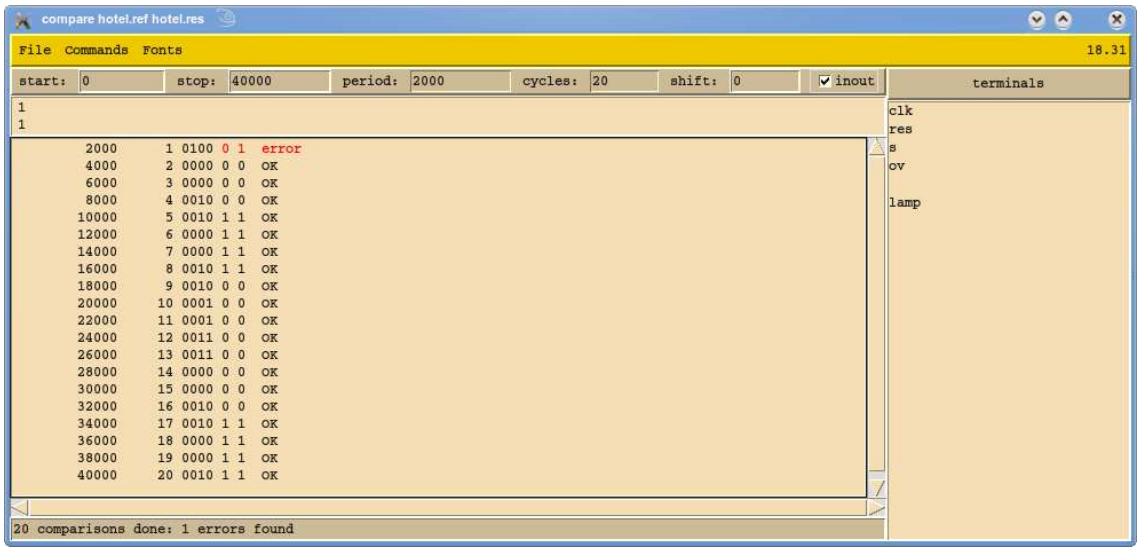
Figuur 12: Het main-window van het programma *GoWithTheFlow* na alle stappen voor het hotel voorbeeld te hebben doorlopen

Een nog wat nauwkeurige manier van verificatie bestaat uit het simuleren van een uit de layout geextraheerde transistor beschrijving. Daartoe moet eerst een geschikte commando file voor de switch-level simulator gemaakt worden. Klik op Generate → Command file en klik vervolgens File → Generate from en selecteer de eerder gemaakte file hotel.lst. Schrijf de gegenereerde commando file weg met File → Write als hotel.cmd. Om te simuleren, klik op de layout rechthoek en selecteer Simulate. Na op Doit geklikt te hebben zal allereerst een extractie plaatsvinden en vervolgens een switch-level simulatie. Klik na afloop op ShowResult om een resultaat te zien zoals in figuur 13. Na inzoomen kunnen hier ook de vertragingstijden voor het uitgangssignaal lamp worden waargenomen.



Figuur 13: Switch-level simulatie resultaat

Hoewel het in dit geval duidelijk te zien is dat de resultaten van de simulatie op transistor niveau overeenkomen met de resultaten van de oorspronkelijke VHDL simulatie, kunnen we beide simulatie resultaten ook nog door het programma *GoWithTheFlow* laten vergelijken. We moeten daartoe eerst een simulatie referentie file (.ref) file aanmaken vanuit de eerder aangemaakte .lst file. Klik daarvoor op in *GoWithTheFlow* op Generate → Reference file. In het nieuwe window dat verschijnt klik op File → Generate from en selecteer hotel.lst. Klik vervolgens op File → Write om hotel.ref weg te schrijven en sluit het window af. Klik daarna op de knop Utilities → Compare in *GoWithTheFlow* om het compare window te laten verschijnen. Gebruik File → Read ref om hotel.ref in te lezen en File → Compare res om hotel.res in te lezen en deze te vergelijken met hotel.ref. De uitgangssignalen zullen hierbij vergeleken worden op tijdstippen vlak voor de volgende opgaande klokflank. Als het goed is zullen er alleen afwijkende uitgangssignalen optreden tijdens de initialisatie periode aan het begin van de simulatie (zie figuur 14)



Figuur 14: Het vergelijken van simulatie resultaten

7.4 Restricties ontwerpsoftware

In deze sectie zullen de restricties worden behandeld, die in acht moeten worden genomen bij het gebruik van de ontwerpsoftware.

- De 'ports' van entities van VHDL-beschrijvingen die moeten worden gesynthetiseerd mogen *alleen* van het type *std_logic* of *std_logic_vector* zijn.
  Dit om te voorkomen dat allerlei conversies moeten worden gemaakt, waarbij wellicht niet steeds van dezelfde testbench gebruik kan worden gemaakt, en ook conversies van VHDL naar sls kunnen mislukken.
- Gebruik voor de namen van signalen, ports en entities *kleine letters* Bij de diverse data-omzettingen, die gedurende het ontwerp plaatsvinden kunnen anders fouten ontstaan.
- De namen die worden gebruikt voor terminals en circuits dienen onderscheidbaar te zijn in de eerste 14 letters.
- Voor entities, architectures en configurations moeten aparte files worden gemaakt.
- Geef de layout cellen dezelfde naam als de overeenkomstige circuit cel. Dit om bij extractie van VHDL-code uit het layout-gedeelte naam-problemen te voorkomen.
- signalen van het type 'inout' mogen alleen worden gebruikt voor 'analoge' signalen, dus signalen die aan de aangesloten moeten worden op een 'direct' buffer aan de rand.
- Aan ports mogen geen gedeelten van een std_logic_vector worden aangesloten.

De synthese-software voor het genereren van de circuits stelt ook beperkingen aan de VHDL-beschrijvingen. Naast de algemene aanwijzingen in appendix E voor het maken van "synthetiseerbare" VHDL-beschrijvingen noemen we nog de volgende aandachtspunten:

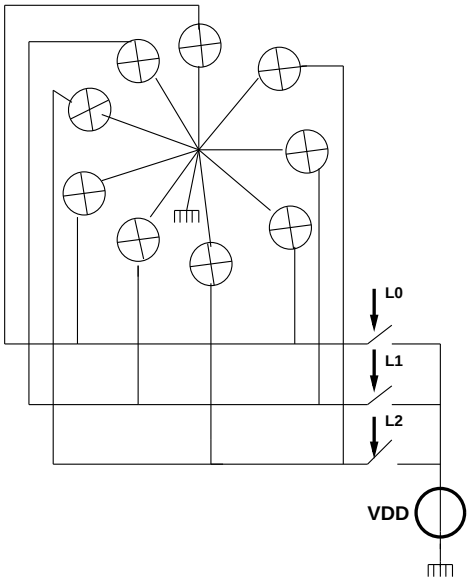
- Initialisaties van signalen bij hun declaratie worden genegeerd.
  Dus bijv. *signal a: std_logic := '0';* wordt gelezen als *signal a: std_logic;*.
- After-clauses in statements worden genegeerd.
  Dus bijv. *a <= '1' after 2 ns;* wordt gelezen als *a <= '1';*.
- Process-statements moeten worden gemaakt met een process-list waarin, bij een register beschrijving, alleen het klok signaal voorkomt, of, bij een beshcrijving van een combinatorische schakeling, *alle* signalen voorkomen die tijdens de uitvoering van het process worden gelezen.
- In case- en select-statements moeten *alle* mogelijke waarden worden behandeld.
- In iedere tak van case- en if-statements moeten *alle* uitgangs-signalen een waarde krijgen.

Om de tijdvertragingen van de schakelingen te kunnen bekijken is de nauwkeurigheid van de sls-simulatie default op 100 ps ingesteld. De maximale tijsduur die dan nog kan worden gesimuleerd, zonder 'overflow'-problemen te krijgen is 100 ms. *Hou hier met het simuleren van de* VHDL *code rekening mee en pas zonodig de klok-frequentie van de schakeling aan.*

7.5 Opdrachten ontwerp van een besturing

7.5.1 De Ron Brandsteder Lights

Om t.v. shows wat meer aanzien te geven blijkt het noodzakelijk om regelmatig wat lampen op en rond het podium aan en uit te zetten, waardoor de inhoud van de show wat minder aandacht behoeft. De lampen waarom het gaat worden gedacht in een cirkel te staan. Het aantal lampen is een veelvoud van drie. Elke derde opeenvolgende lamp is met dezelfde schakelaar verbonden (zie figuur 15). Hierdoor ontstaan drie groepen lampen. Elke groep is door een aparte schakelaar aan of uit te zetten.



Figuur 15: Aansluitschema Ron Brandsteder Lights

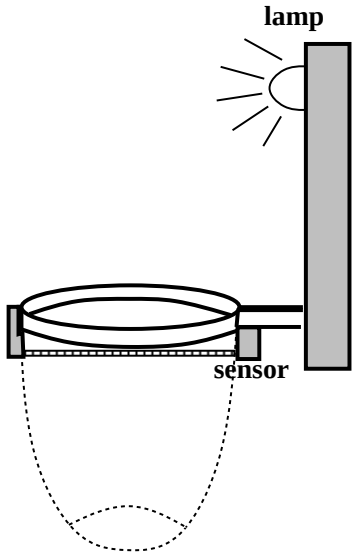
U krijgt de opdracht om een besturing voor deze schakelaars te ontwerpen.
De besturing moet de volgende mogelijkheden hebben:

- Alle lampen staan uit.
- De drie groepen worden rechtsom-draaiend na elkaar aan en uit gezet.
- De drie groepen worden linksom-draaiend na elkaar aan en uit gezet.
- Alle lampen blijven in dezelfde stand staan.

De input van de besturing bestaat uit twee stuursignalen s0 en s1. De output bestaat uit drie outputsignalen L0, L1 en L2, die elk een der schakelaars bedient.

7.5.2 De Basket-aanduiding

Joop van den Beginne heeft voor het nieuwe seizoen weer een nieuw televisiespel bedacht. Bij een bepaald onderdeel moeten de deelnemers een bal in een basketbal-netje gooien. Als het lukt, gaat er een toeter af. Naast de toeter bevindt zich op het bord ook een signaallamp die als het spel begint uit is, gaat knipperen als één keer raak is gegooid en aan gaat als twee keer raak is gegooid. De quiz-master kan met een drukknop de schakeling weer in de begintoestand brengen.



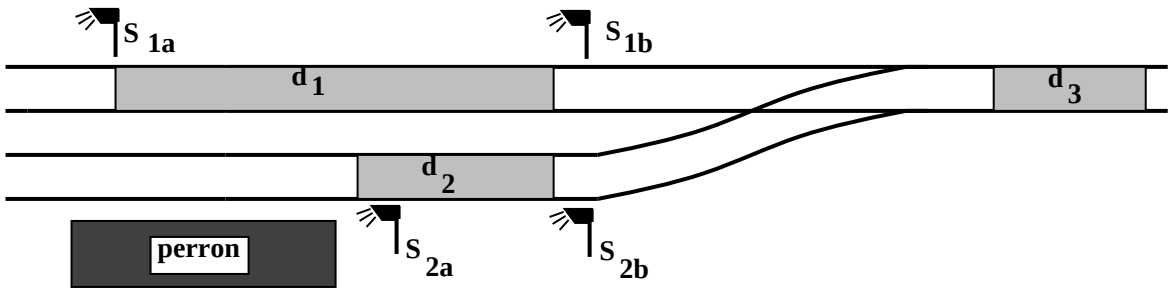
Figuur 16: Situatieschets basket

Aan de basket-rand is een lichtsluis aangebracht die een logische '1' afgeeft als er een bal raak geworpen wordt.
U krijgt de opdracht een schakeling te ontwerpen die aan de gestelde eisen voldoet. Verder geldt er dat:

- De aangeboden klok heeft een frequentie die geschikt is om een lamp te laten knipperen (5 Hz).
- Het sensor- en resetsignaal duren één klokpuls en zijn ontdenderd en gesynchroniseerd (met de klok).
- De toeter moet gedurende minimaal twee klokpulsen hoorbaar zijn.

7.5.3 De Treinbeveiliging

Op het trajekt Madurodam-Lutjebroek wordt een nieuw station gebouwd. Om een goede doorstroom van de sneltreinen te garanderen, wordt bij de perrons een extra spoor aangelegd. Hierdoor blijft het hoofdspoor vrij voor passerende sneltreinen.



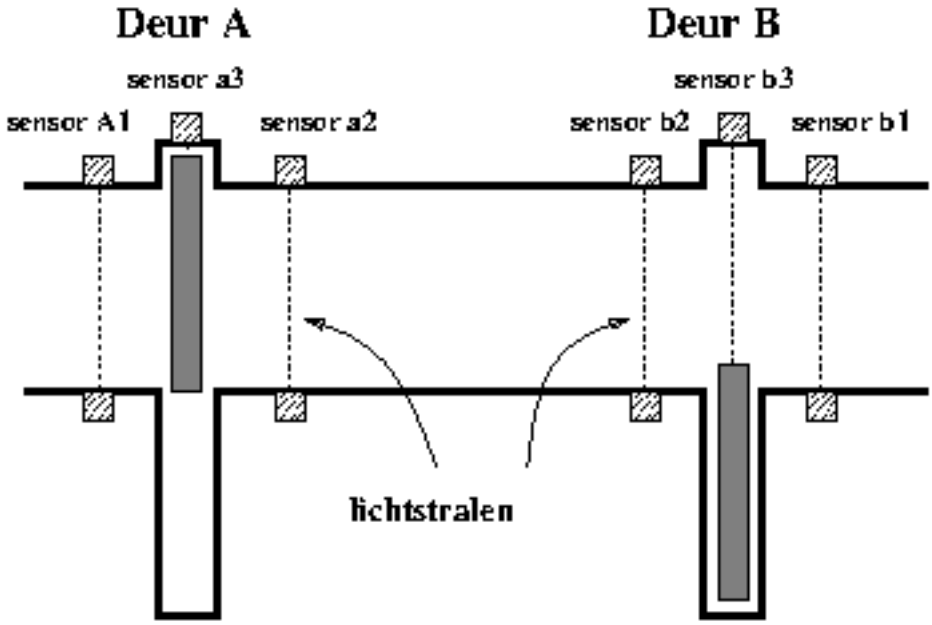
Figuur 17: Situatieschets spoorbaan

Om ongelukken te voorkomen is ook een seinenstelsel aangebracht. Er zijn vier seinen geplaatst ($S_{1a}, S_{1b}, S_{2a}, S_{2b}$) die worden aangestuurd met twee signalen (S_1 en S_2). Als een signaal een logische '1' is, zijn de bijbehorende seinen S_{na} en S_{nb} respectievelijk geel en rood. In het andere geval zijn ze beide groen.
Om de seinen te kunnen besturen, zijn ook drie detectoren (d_1, d_2, d_3) aangebracht. Deze geven een logische '1' af als een trein zich in hun baanvak bevindt.
U krijgt de opdracht een schakeling te ontwerpen die ervoor zorgt dat de seinen een zodanige stand krijgen dat er bij juiste handeling geen ongelukken kunnen gebeuren. Het ontwerp dient tevens zo te zijn dat bij een tegelijk aankomen van zowel een trein op het hoofd- als zijspoor de trein op het hoofdspoor voorrang krijgt.
Verder geldt er dat:

- De remweg van een trein is kleiner dan de afstand tussen S_{na} en S_{nb}.
- Het is niet mogelijk dat een trein twee detectoren tegelijkertijd activeert.
- De treinen rijden op dit baanvak slechts van links naar rechts.

7.5.4 De 'luie' Sluisdeur

De opdracht is de logica te ontwerpen voor twee deuren die als 'anti-tochtschakeling' moeten werken, net zoals de achterdeur van het elektro-gebouw. De configuratie van de 2 deuren A en B is in onderstaande figuur getekend.



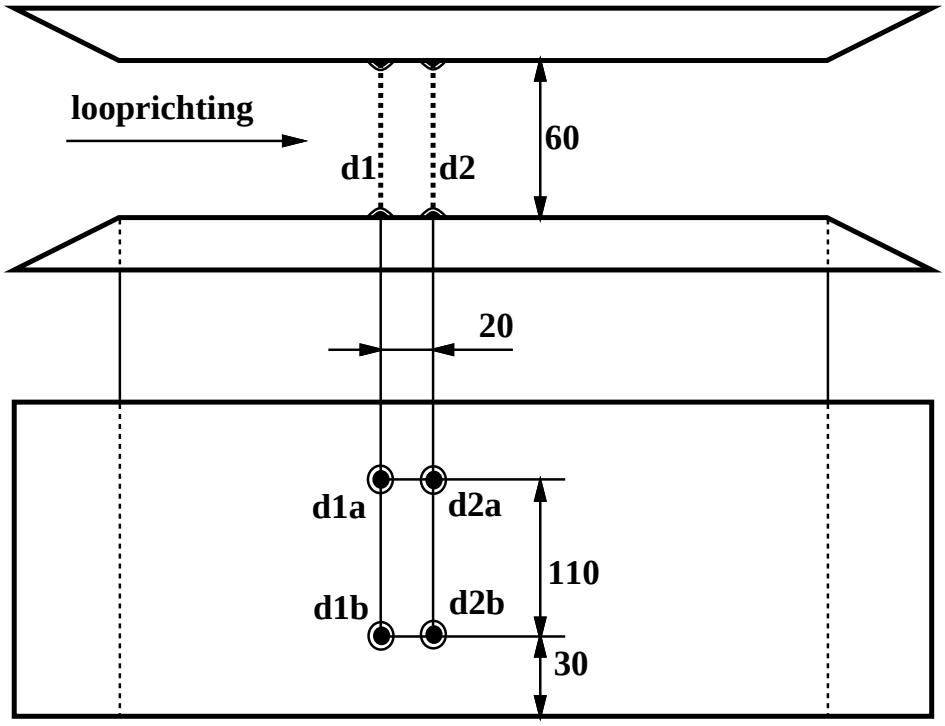
Figuur 18: Situatieschets sluisdeuren

Bij elke deur zijn 3 lichtstraalsensors gemonteerd om de personen te detecteren. Een sensor geeft een logisch '1' signaal af als de lichtstraal wordt onderbroken. De logica voor de deurschakeling moet aan de volgende regels voldoen:

- In elke situatie is er altijd *minimaal* één deur gesloten.
- De deur kan niet dicht als sensor a3 (of b3) een signaal geeft. Dit is om nare ongelukken te voorkomen.
- Personen moeten zowel van links naar rechts als van rechts naar links door de deur kunnen.
- Om slijtage aan het deurmechaniek te beperken is de schakeling 'lui', d.w.z. hij gaat niet vanzelf naar de toestand met 2 gesloten deuren terug.
- Als tegelijkertijd bij deur A en bij deur B een persoon aankomt krijgt de persoon bij deur A voorrang.
- Bij een reset of bij het aanzetten van de stroom moeten beide deuren dicht zijn.

7.5.5 De Personendetector

Bij veel evenementen blijkt het gewenst om na afloop te weten hoeveel mensen er aanwezig geweest zijn. Om dit te kunnen meten is er een vrij smalle (60 cm) gang gemaakt waar maximaal één persoon tegelijk doorheen kan. In deze gang zijn kort achter elkaar twee lichtsluizen (d1, d2) opgenomen die elk weer uit een combinatie van twee infrarood zenders en ontvangers (a, b) bestaan. Als zowel de a als b zender-ontvanger-combinatie onderbroken is, wordt er door de desbetreffende sensor een hoog signaal afgegeven. Signaal d1 wordt dus hoog als zowel d1a als d1b onderbroken zijn.
De looprichting is van links naar rechts. De uitgang is een signaal dat ervoor zorgt dat een teller opgehoogd wordt.



Figuur 19: Situatieschets personendetector

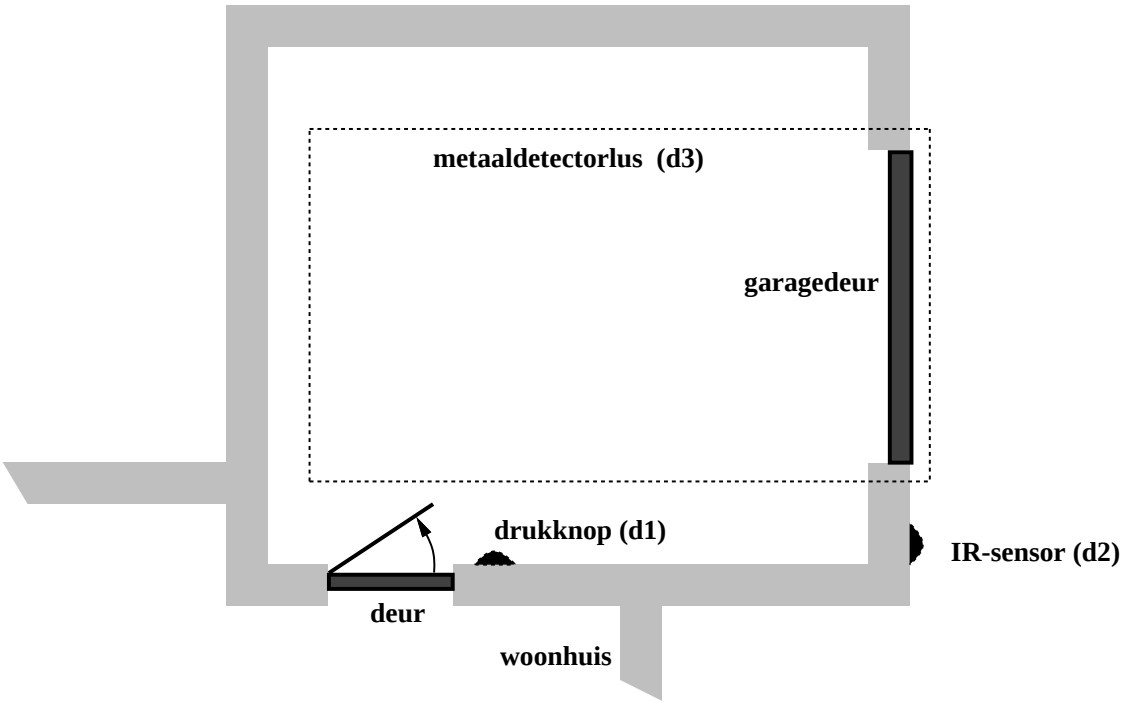
U krijgt de opdracht een besturing te maken die aan bovengestelde eisen voldoet.
Overige voorwaarden:

- Een persoon die in tegengestelde richting loopt, mag niet gedetecteerd worden.
- Het is niet mogelijk om het tel-signaal op de klok-ingang van de teller aan te sluiten.

Facultatief: Maak een schakeling die tevens personen detecteert die van rechts komen en dan een count-down-signaal afgeeft.

7.5.6 De Garagedeur

Om niet elke keer uit z'n auto te moeten stappen om de garagedeur open en dicht te doen, wil de heer B. Modaal een garagedeur besturing laten installeren die ervoor zorgt dat zijn garagedeur automatisch opengaat als hij aankomt en sluit als hij vertrekt.
Naast de deur is daarvoor een infrarood (IR) sensor (d2) aangebracht die een "hoog" signaal afgeeft als hij het signaal herkent dat door een zendertje in de auto uitgezonden wordt. Tevens is in de vloer van de garage een metaaldetectorlus (d3) aangebracht waarmee gedetecteerd kan worden of er een auto in de garage aanwezig is. Als laatste is er bij de deur naar de woning nog een drukknop aangebracht waarmee de garagedeur met de hand bediend kan worden.



Figuur 20: Situatieschets garage

U krijgt de opdracht een besturing te ontwerpen waarmee met de aanwezige sensoren de garagedeur bediend kan worden.
Overige voorwaarden en opmerkingen:

- Als de drukknop wordt ingedrukt moet de garagedeur altijd openen of sluiten.
- Als de IR-sensor geactiveerd wordt, moet de garagedeur alléén openen of sluiten als de garage leeg is.

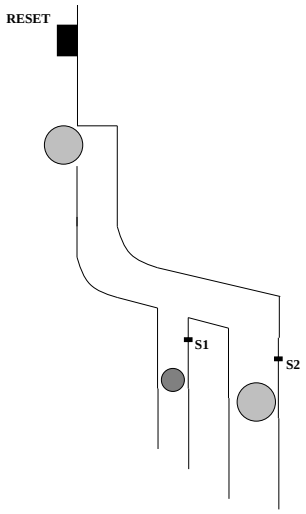
7.5.7 De Frisdrankautomaat

Ontwerp een besturing voor de afhandeling van het ingeworpen geld in een frisdrankautomaat.
De frisdankautomaat kan muntjes van 20 cent en euros verwerken, maar kan niet wisselen. De prijs van een blikje drank is 1,20 euro. Een schets van het transportsysteem is gegeven in figuur 21.

De muntstukken worden na inworp op grootte geselecteerd. De muntjes van 20 cent komen in het eerste kanaal terecht en worden gedetecteerd door sensor S1. De euros komen in het tweede kanaal terecht en worden gedetecteerd door sensor S2. De sensoren geven na detectie een hoog signaal af ter breedte van minimaal een klokpuls.

Indien er 1 muntje van 20 cent en 1 euro is ingeworpen moet er een puls aan een relais R2 worden afgegeven waarmee de frisdrankautomaat een blikje vrijgeeft. Dit relaissignaal hoeft slechts 1 klokpuls lang te zijn. De resetknop zorgt ervoor dat het tot dan toe ingeworpen geld voor een nieuw blikje weer teruggegeven wordt. Dit wordt bewerkstelligd door een hoog signaal op een relais R1. Tevens moet relais R1 worden aangestuurd indien er voor een nieuw blikje 2 muntjes van 20 cent of 2 euros zijn ingeworpen. D.w.z. dat bij foutieve inworp al het ingeworpen geld wordt teruggegeven. In figuur 21 zijn de relais R1 en R2, die resp. het teruggeven van het geld en het vrijgeven van een blikje besturen, niet weergegeven.

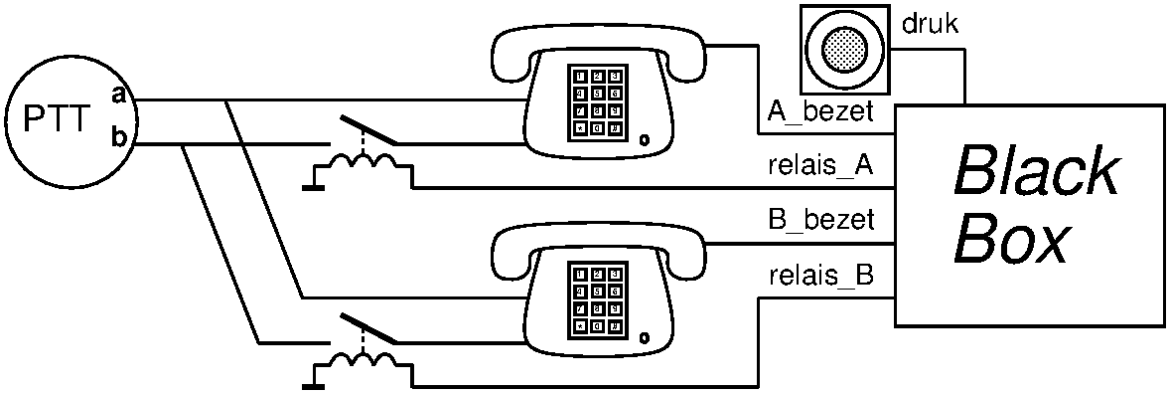
Facultatief: Bedenk een uitbreiding van het systeem waarbij ook de inworp van 6 muntjes van 20 cent is toegestaan en/of alleen het teveel ingeworpen geld wordt geretourneerd. De eventueel benodigde extra uitgangssignalen kunnen niet naar buiten worden uitgevoerd, maar kunnen in de simulaties wel getest worden.



Figuur 21: Schets geld-inwerpsysteem frisdrankautomaat

7.5.8 De Telefoonschakeling

Een student, die nog niet zo ver is met z'n studie, heeft het volgende probleem: Hij deelt samen met een huisgenoot de telefoonaansluiting. Tot nu toe stonden de telefoons parallel aangesloten, maar ze vinden het vervelend dat ze door elkaar heen kunnen praten, wanneer ze beiden tegelijk de telefoon opnemen. Degene die opbelt, begrijpt er dan niets meer van.



Figuur 22: Schets schakeling telefoons

De studenten hebben bedacht dat ze het probleem met de bovenstaande schakeling kunnen verhelpen. De telefoons zijn nu beide via een relais aangesloten op de telefoonlijn. Daarnaast heeft toestel A een drukknop, waarmee doorverbonden kan worden. U krijgt de opdracht om de 'black box' in te vullen met een schakeling, die moet voldoen aan de volgende eisen:

- Wanneer opgebeld wordt, moeten beide telefoons rinkelen.
- Wanneer één van beiden de telefoon opneemt, wordt het andere toestel van de lijn afgeschakeld.
- Wanneer beiden tegelijk de telefoon opnemen, krijgt toestel A voorrang.
- Als de drukknop wordt ingedrukt, worden beide toestellen met de telefoonlijn verbonden. Op deze manier kunnen de studenten de opbeller doorverbinden naar het andere toestel.

De input van de schakeling bestaat uit drie signalen. De signalen A_bezet en B_bezet geven een logische '1' af als de hoorn van het betreffende toestel niet op de haak ligt. Het signaal druk geeft een logische '1' af als de drukknop wordt ingedrukt. De uitgangssignalen relais_A en relais_B zorgen ervoor dat de toestellen met de telefoonlijn worden verbonden. Daarnaast hebt u de beschikking over voedingssignalen en een kloksignaal (niet getekend in de figuur). De schakeling heeft geen apart resetsignaal.

8 De Groepsopdracht

8.1 Inleiding

In dit hoofdstuk worden beschrijvingen gegeven van de mogelijke hoofdopdrachten:

- de Dialmemo,
- de Avant-Garde Klok,
- de Infrarood-Besturing,
- de Reactiemeter,
- de Pong Spelcomputer,
- Alternatieve opdracht.

Eén van deze opdrachten zult u met uw groep gaan ontwerpen.
Naast de beschrijvingen van de opdrachten vind u ook bij elke opdracht een aantal specificaties waaraan de schakelingen moeten voldoen. Deze specificaties zijn niet volledig, maar bedoeld als eerste aanzet. Zelf zult u nauwkeuriger moeten specificeren.
Nadat de opdrachten afzonderlijk zijn beschreven worden er er vervolgens nog enige algemene randvoorwaarden aangegeven waar u bij het ontwerpen terdege rekening mee dient te houden.
Tenslotte volgen nog een paragraaf met enige adviezen over hoe u de systeemspecificatie het best aan kunt pakken, een paragraaf over de beschikbare ruimte op de chip, een paragraaf die beschrijft hoe tijdens het ontwerpen al rekening moet worden gehouden met het testen dat later plaatsvindt, en een paragraaf over de afronding van het ontwerp.
Voor analoge deelschakelingen die mogelijk voorkomen in het ontwerp wordt men verwezen naar appendix F.

8.2 De Dialmemo

8.2.1 Inleiding

Beeld u eens in dat u werkt bij de firma TECHNOGADGET. Zoals de naam al suggereert zit uw bedrijf in de relatiegeschenkenbranche. Een aantal jaren geleden vond de oprichter van TECHNOGADGET (een oud-TU-student) de alom bekende ”keyfinder” uit. De keyfinder is een groot model sleutelhanger (met firma-opdruk) die gaat piepen bij een plotseling hard geluid (b.v. fluiten). Het succes van dit idee was zo enorm dat deze student z’n lelijke eend al spoedig kon inwisselen voor een glimmende Porsche.
Elke doorgewinterde beursbezoeker heeft ondertussen al hele sleutelbossen vol keyfinders. De markt raakt daardoor een beetje verzadigd en bovendien is het nieuwtje er langzamerhand een beetje van af. Als gevolg daarvan gaan de zaken voor TECHNOGADGET momenteel slecht. Het is duidelijk dat het bedrijf snel een nieuw produkt nodig heeft, anders moet binnenkort de lelijke eend weer van stal gehaald worden.

Op een blauwe maandagmorgen kreeg de directeur weer een briljant idee: de DIALMEMO. Het is weer een groot model sleutelhanger, maar deze keer bevat hij een heel klein telefoontje. De DIALMEMO bestaat uit een toetsenbord van een druktoets-telefoon, een chipje, een luidsprekertje en een batterij, alles in een klein elegant huisje met firma-opdruk. Door het apparaatje tegen de microfoonkant van een telefoonhoorn te houden en vervolgens op een knop te drukken, wordt het firmanummer automatisch gebeld. Ook is het met de DIALMEMO mogelijk om een draaischijf-telefoon als druktoets-telefoon te gebruiken. Technisch is het principe eenvoudig. Moderne telefoon-centrales werken met een toonkies-systeem. Hierbij wekt elke knop van een druktoets-telefoon een specifieke toon-combinatie op die in de centrale herkend wordt. In plaats van de toetsen één voor één in te drukken, fluit de DIALMEMO het nummer rechtstreeks in de microfoon van de hoorn. Door de eenvoudige opbouw moet de DIALMEMO voor rond de 1 euro per stuk in China te produceren zijn.
Het is duidelijk dat dit een revolutionaire vinding is. Zo kan bijvoorbeeld de slogan ”even Apeldoorn bellen” een volledig nieuwe dimensie krijgen. Iedere verzekerde krijgt nu als welkomsgeschenk een DIALMEMO met daarin opgeslagen het nummer in Apeldoorn. Centraal Beheer Verzekeringen was waanzinnig enthousiast!
Voor allerlei andere bedrijven die sterk van de telefoon afhankelijk zijn, is de DIALMEMO een uitkomst. Vooral voor 06-nummers ligt een enorme markt braak. Zo zou de exploitant van ”Harry’s gay box” (een homo-babbelbox) er al over denken om een DIALMEMO (met daarin zijn nummer) gratis toe te sturen aan alle abonnees van de gay-krant. Bij gemiddeld 15 minuten babbelen (à 50 cent per minuut) per DIALMEMO is de exploitant al uit de kosten!
U krijgt als groep de opdracht van TECHNOGADGET om de chip te maken die het hart vormt van de DIALMEMO.

8.2.2 Specificaties Dialmemo-chip

- Als invoer dient een toetsenbord, een losse druktoets en een aan/uit-schakelaar. Het toetsenbord heeft de bekende telefoon-layout, d.w.z. de cijfers 0 t/m 9, # en *. De laatste toetsen dragen in ons geval resp. de namen ”read” en ”dial”. De losse toets draagt de naam ”store”.
  Omdat het toetsenbord en ook de losse druktoets buitengewoon low-cost zullen zijn, moet er rekening gehouden worden met het ”denderen” van de contacten. Dit ”denderen” houdt in dat de impedantie-verandering bij het indrukken van een toets niet monotoon verloopt. In de paragraaf ”Overige Specificaties” wordt hier verder op ingegaan. De toetsen van het toetsenbord zijn in een 3x4-matrix geschakeld.
  De uitgangen van het toetsenbord en de losse druktoets worden direct verbonden met de pinnen van de chip.

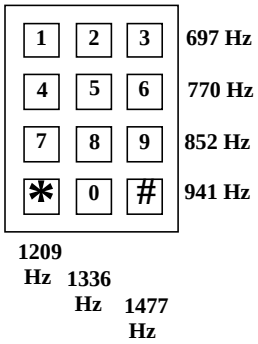
- Aan de uitgang van de chip verschijnen toon-combinaties volgens de officiële CCITT-norm voor telefonie. Zie hiervoor de ”Overige Specificaties”.

- Om de DIALMEMO als gewone telefoon te gebruiken, dient eerst een nummer ingetoetst te worden, waarna door de dial-toets (*) te drukken, dit nummer afgespeeld wordt. Als vervolgens nogmaals dial wordt gedrukt, wordt het laatst ingetoetste nummer opnieuw afgespeeld. Op deze manier is er dus een ”redial”-mogelijkheid.
  Dit ”dial-geheugen” dient maximaal 16 cijfers te kunnen opslaan.
  FACULTATIEF: Bij intoetsen van het nummer dienen de betreffende toon-combinaties hoorbaar te zijn.

- De schakeling heeft naast het dial geheugen ook een ”vast” geheugen dat ook een telefoonnummer van 16 cijfers kan opslaan. Dit ”store” geheugen kan worden geprogrammeerd door de (losse) store toets in te drukken na invoer van een nummer. Oproepen van dit geheugen geschiedt door de read toets te drukken.
  Het is de bedoeling dat de store toets niet in te drukken is door Jan de Beursbezoeker.
- De DIALMEMO is een wegwerpartikel en zal geen verwisselbare batterij hebben. Toch moet het apparaat enkele jaren mee kunnen gaan. Hiervoor is vooral in rusttoestand een laag stroomverbruik noodzakelijk.
- De inhoud van zowel het dial als het store geheugen mag niet verloren gaan wanneer het apparaat zich in rusttoestand bevindt of geactiveerd wordt.
- FACULTATIEF: De schakeling moet bestand zijn tegen het tegelijkertijd indrukken van meerdere toetsen.
  Bedenk zelf wat er moet gebeuren als een telefoonnummer wordt ingetoetst dat langer is dan 16 cijfers.

8.2.3 Overige Specificaties

CCITT-norm voor telefonie Bij het zgn. Dual-Tone Multiple-Frequency (DTMF)-systeem voor telefonie hoort bij elke toets een combinatie van een lage en een hoge toon. Dit is schematisch in figuur 23 aangegeven.



Figuur 23: Frequentie verdeling

Hiervoor gelden de volgende specificaties:

gebruikte frequenties:	laag:	697, 770, 852, 941 (Hz)
	hoog:	1209, 1336, 1477 (Hz)
onnauwkeurigheid:	< 1.5 %	
vermogensverschil tussen tonen:	< 6 dB	
signaalduur:	> 40 ms	< 2 s
signaalpauze:	> 70 ms	< 2 s
signaal(duur+pauze):	> 120 ms	< 2 s
signaalvermogen:	100 μW ($\pm$ 30 %)	
harmonische distorsie:	< 0.01 of -20 dB	

Kristal-oscillator De onnauwkeurigheid van een kristal-oscillator is kleiner dan 1 %.

Toetsenbord

vorm:	3x4-matrix	
impedantie:	uit:	> 10 MΩ
	aan:	< 100 Ω
stabiliseringstijd:	< 50 ms	

Luidspreker

impedantie:	50 Ω ($\pm$ 10 %)
resonantie-frequentie:	440 Hz
bandbreedte:	350-5000 Hz
vermogen:	< 0.3 W
rendement:	100 % (aanname)

8.3 De Avant-Garde Klok

8.3.1 Inleiding

Klokken zijn behalve technische hoogstandjes ook altijd mode-objecten geweest. Iedere periode in de geschiedenis kent zijn eigen karakteristieke uitvoering van een tijdmeter. Het ontwerpen van zoiets "gewoons" als een klok kan daarom nog heel bijzonder zijn. De informatie-overdracht van klok naar mens kan op talloze manieren verzorgd worden. Dit loopt van de ouderwetse wijzerplaat met wijzers tot aan de elektroden op de grote hersenen met een bionisch RS232-interface. Puur technisch gezien wordt van klokken een steeds grotere nauwkeurigheid verlangd. Dit alles in beschouwing genomen blijft het nog steeds een uitdaging om een klok te ontwerpen die behalve als functioneel technisch object ook zijn plaats in de wereld verdient door zijn uitvoeringsvorm. De directeur van TECHNO-GADGET had op een druilerige zondagmiddag het idee gekregen om een klok met een ultieme nauwkeurigheid te maken. Hij dacht ineens aan een zender die in Duitsland staat, die met atoomnauwkeurigheid de tijd gedecodeerd uitzendt, de zogenaamde DCF-zender. Na verdere ontwikkeling van zijn ideeën, heeft hij contact opgenomen met de beroemde Italiaanse designer Verstamo, om een klok te ontwerpen die eeuwigheidswaarde heeft.

Er zijn talloze manieren om de tijd zichtbaar te maken. De ontwerper heeft gekozen moderne middelen te gebruiken, om de dynamiek van de techniek op de bezitter van de klok over te dragen. Hiervoor gebruikt hij een LED-display voor de aanduiding van de uren, zeven LEDs voor de aanduiding van de weekdag en twee draaispoelmeters voor de aanwijzing van de minuten en seconden. Om de gebruikers te helpen de klok zo te plaatsen dat deze het DCF-signaal kan ontvangen, is ook op een

strategische plaats een LED aangebracht, die aangeeft of het DCF-signaal wordt ontvangen. Zo mogelijk zal de klok ieder kwartier het wijsje spelen dat op dat moment ook door de Big Ben in Londen wordt gespeeld. Speciaal voor mensen die ooit in Londen geweest zijn maakt dit de klok natuurlijk extra aantrekkelijk.

Helaas heeft Verstamo geen verstand van elektronica, dus de chip die dit alles moet besturen zal door de echte deskundigen ontworpen moeten worden. Hier is natuurlijk haast bij, want zoals alle leken denkt ook Verstamo dat elektronisch alles zo maar kan. Hij heeft zijn klok dus al verkocht aan een grote fabrikant die reeds een grote reclamecampagne heeft opgezet. Verstamo kan dan ook een claim van ongeveer 100 miljoen euro aan geïnvesteerd reclamegeld verwachten indien het ontwerp niet op tijd komt. Zijn adviseurs hebben hem al gewaarschuwd dat simulaties ter verificatie van de chip veel tijd kunnen gaan kosten. Wordt bijvoorbeeld een schakeling gesimuleerd met een klokfrequentie van 6 MHz waaruit een puls per week moet komen, dan moet hij beseffen dat de simulator niet veel meer dan 6 miljoen gebeurtenissen per seconde rekentijd kan simuleren en dat de simulatie dus ook makkelijk een week zal kunnen gaan duren. Verder zitten er in een week bijzonder veel seconden, zodat data-opslag voor de simulator ook wel eens een probleem zou kunnen gaan worden. Ook later, wanneer de chip uit de fabriek komt zal het ondoenlijk zijn de chip een week lang in een tester te stoppen om te kijken of de weekaanduiding werkt. De chip zal niet op een veel hogere klokfrequentie kunnen werken, dus hier zal weinig winst te behalen zijn.
Het is daarom zaak ruime aandacht te besteden aan simuleerbaarheid en testbaarheid en niet aan een grote hoeveelheid (ontestbare) elektronische grapjes. Het aanbrengen van verschillende testmogelijkheden zoals het “opdelen” van delerketens die los testbaar zijn binnen een redelijke tijd is van zeer groot belang.

8.3.2 Specificaties DCF-ontvanger

- Als invoer dient een antenne-signaal, dat de DCF-code bevat en door de al bestaande ontvanger wordt omgezet naar een signaal dat direct als invoer kan dienen voor de schakeling, die voor de verwerking van dit signaal zorgt. Deze signalen hebben logische niveaus die door deze schakeling verwerkt kunnen worden, maar nog niet gesynchroniseerd en gedecodeerd zijn.

- Als uitgang van de DCF-klok dienen enige displays die de tijd en ook een luidspreker die de tonen van de Big Ben melodie hoorbaar maakt. Bij het genereren van deze tonen kunnen verschillende luxe-klassen worden onderscheiden. De tonen kunnen een verschillende tijdsduur hebben, maar ook een volume dat in de tijd afneemt. Dit geeft het effect of een echt klokkenspel wordt bespeeld. Deze opties brengen wel een ingewikkelder en groter ontwerp met zich mee!

- De nauwkeurigheid van de klok wordt geleverd door de beroemde DCF-tijdzender. Deze zender zendt op 77.5 kHz een signaal uit in de vorm van een serie bits. Elke minuut wordt er een reeks van 59 bits uitgezonden die de complete tijd weergeeft (tijd, dag, maand, jaar en controlebits). De afwijking is ongeveer 1 seconde per 300.000 jaar, dus dat is net voldoende. Een bijzonder simpele ontvanger (die niet ontworpen hoeft te worden) levert aan de uitgang een digitaal signaal (pulsen van 100 ms en 200 ms), waaruit de tijd bepaald kan worden.

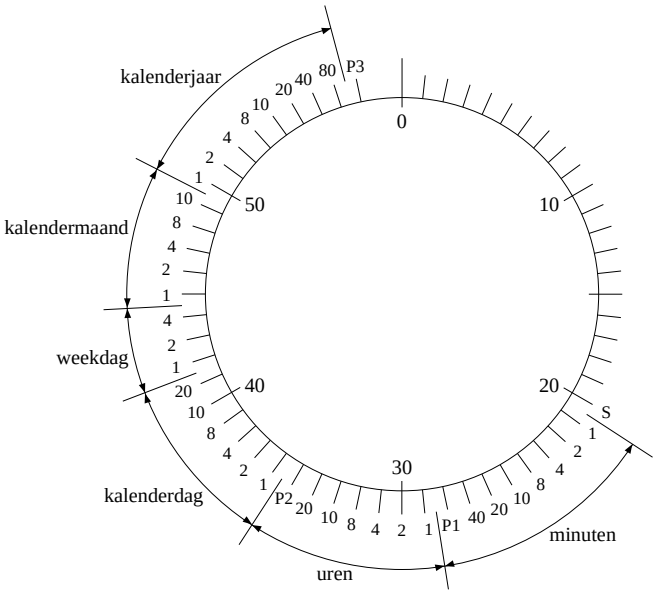
- De ontvangst van het DCF-signaal zal doorgaans goed zijn. Als de zender echter niet ontvangen wordt, of als er storingen worden gedetecteerd, dan zal de klok moeten besluiten het voorlopig maar met het eigen kristal te doen. Zodra goede ontvangst weer mogelijk is, zal de

klok zich weer precies gelijk zetten. Wanneer een DCF-signaal wordt ontvangen, moet de klok dit aangeven door het laten branden van een LED.

- De firma TECHNOGADGET verwacht dat de eerste serie exclusieve klokken nog voor grote bedragen verkocht kunnen worden. Wanneer de klokken bekend worden, wil iedereen zo'n klok kopen. Om iedereen in staat te stellen zo'n klok te kopen, moeten ook goedkopere versies gemaakt kunnen worden. Door verschillende mogelijkheden niet te gebruiken, kan een goedkope versie worden aangeboden. Door een modulair ontwerp te maken, is het eenvoudig om bepaalde onderdelen weg te laten. Dit heeft als voordeel dat de verschillende circuitdelen los van elkaar getest kunnen worden, omdat ze ook los van elkaar kunnen werken. Dit is ook prettig voor de fabrikant, omdat deze bij uitval tijdens de fabricage, wanneer bijvoorbeeld het speelwerk defect is, de klokken kan gebruiken in een wat goedkopere versie van de DCF-klok zonder geluid.

8.3.3 Overige specificaties

DCF-signaal De codering van het DCF-signaal ziet er als volgt uit:

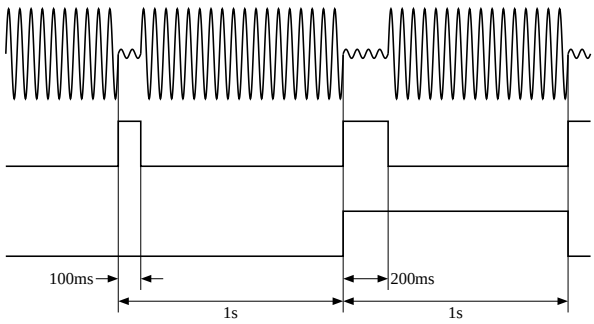


Figuur 24: Codering van het DCF-signaal

- Alle waarden worden gegeven in BCD-code.
- Er zijn drie pariteitsbits (controle-bits: P1, P2, P3) aanwezig. Deze kunnen genegeerd worden.
- Bij de weekdag is maandag als "001" gecodeerd. De overige coderingen zijn triviaal.

Het signaal zelf ziet er ongeveer zo uit als bovenaan aangegeven is in figuur 25

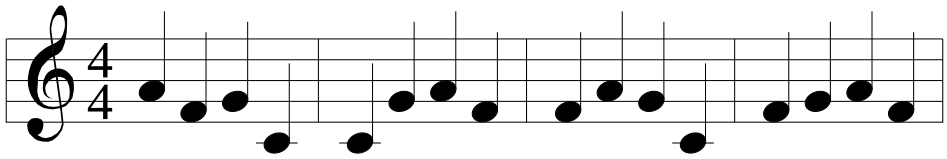
Een (overigens vrij eenvoudige) ontvanger, die u niet zelf hoeft te maken, maakt er een signaal van dat lijkt op het middelste signaal. Dit signaal wordt aangeboden aan de chip.



Figuur 25: Vorm van het DCF-signaal

De DCF-zender genereert om de seconde een puls met een breedte van 100 ms (voor een ’0’) of 200 ms (voor een ’1’). Op de 60-ste seconde wordt geen puls uitgezonden.
Het onderste signaal geeft aan hoe u dit zou kunnen decoderen.

Big Ben melodie de Big Ben melodie ziet er als volgt uit:



Figuur 26: Big Ben melodie

De toonhoogten die gebruikt worden zijn:

C	1046.5 Hz
F	1396.9 Hz
G	1567.9 Hz
A	1760.0 Hz

Draaispoelmeter De draaispoelmeters zijn lineair en vragen voor een volledige uitslag een stroom van 2 mA. De impedantie is kleiner dan 100 Ω.

Uitgangen De uitgangen van de DCF-klok kunnen bij een te grote belasting gebufferd worden. Om een goede aansturing van de buffers mogelijk te maken, moet de uitgangsstroom van de (analoge) signalen (extern) regelbaar zijn.

Toetsen

impedantie:	uit:	> 10 MΩ
	aan:	< 100 Ω

stabiliseringstijd:	< 50 ms

Luidspreker

impedantie:	50 Ω ($\pm$ 10 %)
resonantie-frequentie:	440 Hz
bandbreedte:	350-5000 Hz
vermogen:	< 0.3 W
rendement:	100 % (aanname)

Functies Er zijn een aantal duidelijk te onderscheiden functies in de klok waardoor de firma TECHNOGADGET in staat is deelontwerpen uit te besteden aan verschillende teams:

- De DCF-decoder, die in staat is de dcf-pulsen te decoderen en om te zetten in logische niveaus.
- De schakeling die besluit of er een geldige datareeks is binnengekomen uit de ontvanger.
- Een “autonome klok” die parallel de DCF-data kan laden als deze geldig blijkt te zijn, maar die zelfstandig kan lopen in andere gevallen.
- Een display sectie, die de LEDs en de draaispoelmeters kan aansturen.

8.4 De Infrarood-Besturing

8.4.1 Inleiding

Een instituut beschikt over een ionenbron, die ingesteld wordt m.b.v. een vijftal parameters, die alle kunnen worden ingesteld door een getal tussen de 0 en 99. Deze ionenbron is geplaatst in een afgeschermde ruimte omdat hij op een hoge spanning moet zijn aangesloten. Probleem is nu, dat steeds wanneer voor een proef met deze bron een bepaalde parameter moet worden gewijzigd, de gehele opstelling moet worden uitgeschakeld om de betreffende parameter te kunnen instellen. Gelukkig voor dit instituut liep er een slimme TUD-student stage, die het idee opperde om een infrarood kanaal te maken, waarmee de waarden voor de parameters naar de bron-opstelling zouden kunnen worden overgebracht, zonder de opstelling uit te schakelen. Hij merkte op dat de devices voor het instellen van de parameters reeds een busaansluiting hadden, waarop, in binaire code de waarde van de parameter kon worden geplaatst, waarna deze waarde werd overgenomen wanneer de enable lijn van het betreffende device gedurende tenminste 100 us ’hoog’ werd. Bovendien wist hij (hij had het ontwerppracticum met goed gevolg doorlopen en toen gepoogd een dialmemo te maken), dat er van die handige telefoon- toetsenbordjes bestaan, die heel goed als invoermedium van gegevens kunnen dienen.
Helaas, toen dit allemaal bedacht en besproken was met de directie van het instituut (waar veel tijd in ging zitten), was zijn stagetijd om. Maar gelukkig kan een nieuwe op-goep uitkomst brengen door dit infrarood kanaal te realiseren, door een zend- en een ontvang-chip voor dit probleem te maken met de onderstaande specificaties.

8.4.2 Specificaties infrarood-kanaal

- Het infrarood-kanaal zal bestaan uit twee chips: een zender-chip, die een pulsreeks uitzend om de parameters in te stellen, en een ontvanger-chip die de uitgezonden pulsreeks decodeert en de parameters de juiste waarden geeft.

- Als invoermedium voor het kanaal zal gebruik worden gemaakt van een telefoon toetsenbord, waarbij de volgende afspraken gelden:

  - De nummers op het toetsenbord worden gebruikt om de waarden van de parameters op te geven.
  - De *-toets wordt gebruikt om het device waarvan de parameter moet worden veranderd aan te geven. Steeds wanneer op deze toets wordt gedrukt zal het volgende device worden aangewezen.
  - De #-toets wordt gebruikt om de opgegeven waarde van de parameter in het aangegeven device te laden.

  Bovendien dient er rekening mee te worden gehouden, dat het toetsenbord een ontdendertijd heeft van 40 ms.

- De zender moet tevens een uitgang hebben, waarop een 7-segment display kan worden aangesloten, waarop de laatst ingedrukte toets is zichtbaar gemaakt. Hierbij moet de *-toets worden getoond als een 'A' (advance) en de #-toets als een 'U' (update).

- De ontvanger moet een 7-bits brede uitgang hebben, waarop de 7-bits binaire code voor de parameter-waarde wordt geplaatst en een uitgang voor het enable bit van ieder device, waarmee de parameter-waarde in dit device kan worden geplaatst.

- De laatst verstuurde waarden van elk device, dit zijn dus de waarden waarop de bron staat ingesteld moeten worden zichtbaar gemaakt op een LCD. Voorts moeten hierop ook de nieuw in te stellen waarde worden getoond, alsmede een indicatie voor het geselecteerde device waarop deze waarde zal worden geplaatst.
  Hiertoe is een LCD aanwezig (met documentatie), waarop 2 regels van elk 16 karakters kunnen worden weergegeven.

- Daar de zender-chip en ontvanger-chip alleen met elkaar communiceren via het infrarood-kanaal en er verder geen verbinding tussen de twee chips mogelijk is, wordt er dus ook voor het zend- en ontvang-gedeelte gebruik gemaakt van een afzonderlijke klokpuls.
  Er moet dus rekening mee worden gehouden, dat deze twee klokpulsen geen fase-koppeling hebben. Ook moet er rekening worden gehouden met een eventueel verschil in frequentie van beide pulsen van maximaal 5%.

- De pulsduur van de enable-signalen van de devices moet minimaal 100 us bedragen, en gedurende minimaal 50 us voor deze puls tot 50 us na deze puls moet de waarde op de parameter-uitgangen stabiel zijn.

8.5 Een Reactiemeter

8.5.1 Inleiding

Tijdens een feestelijke bijeenkomst raakten een drietal studenten in gesprek over hun uiterst snelle reactietijd op een gebeurtenis (zelfs na een aantal glazen bier). Natuurlijk vond elk van de drie dat hij toch verreweg de kortste reactietijd had. De discussie liep hoog op, tot één van de studenten op het idee kwam om een schakeling te maken, die voor eens en altijd kon uitmaken wie van hen werkelijk het snelst was. Ze bedachten hierna zo'n schakeling die de hieronder beschreven werking zou moeten hebben. Helaas is het tot een realisatie van de schakeling nooit meer gekomen. Maar hier kan een op-groep wellicht uitkomst brengen.

8.5.2 Werking reactiemeter

De reactiemeter moet twee cijfers weergeven: een in te stellen referentiecijfer en een steeds veranderend reactiecijfer. Voorts moet voor elke deelnemer een drukknop aanwezig zijn en een tijdweergave van hun reactietijd.
Wanneer het referentiecijfer en het reactiecijfer gelijk zijn moeten de deelnemers zo snel mogelijk hun drukknop indrukken. Wie het eerst heeft gedrukt behoud zijn tot dan toe gebruikte reactietijd. Bij de twee andere deelnemers moet het verschil tussen hun reactietijd en de reactietijd van de snelste deelnemer bij hun totale reactietijd worden opgeteld. Wie zijn toegestane totale reactietijd (bijv. 1 seconde) heeft gebruikt valt af. Degene die overblijft is de winnaar.
Om te voorkomen, dat 'slimme' deelnemers constant hun drukknop indrukken of ingerukt houden moet ook een straftijd aan de reactietijd van een deelnemer worden toegevoegd, wanneer hij zijn drukknop indrukt terwijl de cijfers niet gelijk zijn.

8.5.3 Specificaties reactiemeter

In verband met het bovenstaande moet de reactiemeter voldoen aan de volgende specificaties:

- Er moet een in te stellen referentiecijfer worden weergegeven.
- Er moet een reactiecijfer worden gemaakt en getoond, dat om de seconde (pseudo-)random verandert.
- Er moet een drietal drukknoppen aanwezig zijn waarop moet worden gedrukt wanneer beide cijfers gelijk zijn.
- Er moet een drietal tijdweergaven zijn, waarop de geaccumuleerde reactietijden van de drie deelnemers zichtbaar zijn.
- Voor elke deelnemer is een indicatie aanwezig, die aangeeft wanneer een deelnemer zijn totale reactietijd heeft overschreden.
- Er moet een reset knop eenwezig zijn, waarmee de schakeling kan worden gereset.
- De reactietijden moeten worden gemeten met een nauwkeurigheid van 0.01 seconde.

- Voor de klok van de schakeling mag gebruik worden gemaakt van een externe pulsgenerator.
- Voor de totale reactietijd die mag worden gebruikt moet een geschikte waarde in de orde van grootte van 1 seconde worden gekozen.
- De 'straf' voor het foutief indrukken van de drukknop mag door de ontwerper(s) van de schakeling worden bepaald.

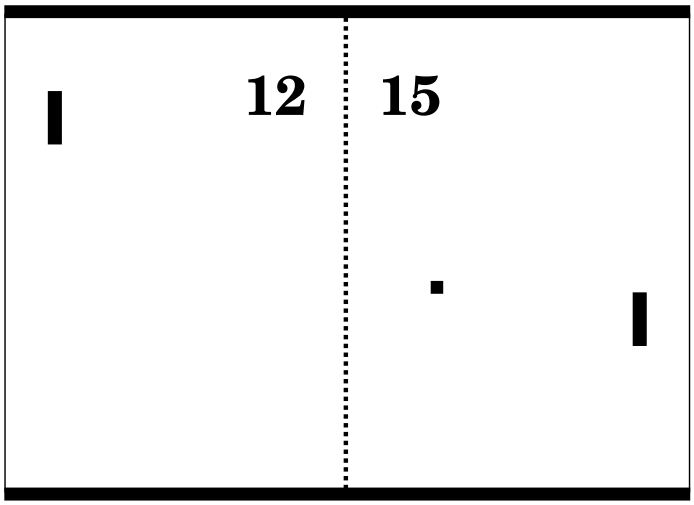
Voor de weergaven van de genoemde grootheden kan worden gekozen uit twee opties:

- Voor het reactiecijfer en het referentiecijfer wordt gebruik gemaakt van een 7-segments display, de tijdsoverschrijding wordt aangegeven met een led en de waarden van de reactietijden, worden via (op de chip te ontwerpen) digitaal-analoog omzetters door draaispoelmeters aangegeven.
- Alle grootheden worden getoond op een lcd-schermpje met een afmeting van 2 regels met elk 16 karakters.

8.6 De Pong Spelcomputer

8.6.1 Inleiding

De opdracht is om op een Sea-Of-Gates chip een spelcomputer te maken waarvan de uitgangen kunnen worden aangesloten op de VGA-ingang van een monitor en waarmee een spel gespeeld kan worden dat populair was in de begintijd van de spelcomputers: "pong tennis". Zie de afbeelding hieronder. Voor meer achtergrondinformatie over "pong", zie ook http://www.pong-story.com/intro.htm.



Figuur 27: Het beeld van de Pong spelcomputer

8.6.2 Het spel

Het idee van het spel is dat twee spelers hun racket omhoog en omlaag bewegen om een balletje te laten terugkaatsen dat zich schuin over het speelveld beweegt. Het balletje voert volkomen elastische botsingen uit tegen de boven en onderwand. Ook tegen de rackets worden volkomen elastische botsingen uitgevoerd, echter met het verschil dat wanneer de bal het racket raakt op de onderste helft de bal altijd terugkaatst naar beneden en wanneer de bal het racket raakt op de bovenste helft de bal altijd terugkaatst naar boven. Elke keer wanneer het balletje botst tegen een wand of racket is het geluid "pong" te horen. Wanneer een speler de bal mist en de bal achter hem passeert krijgt de tegenstander 1 punt erbij. Wanneer één van de spelers 15 punten heeft, heeft deze speler het spel gewonnen.

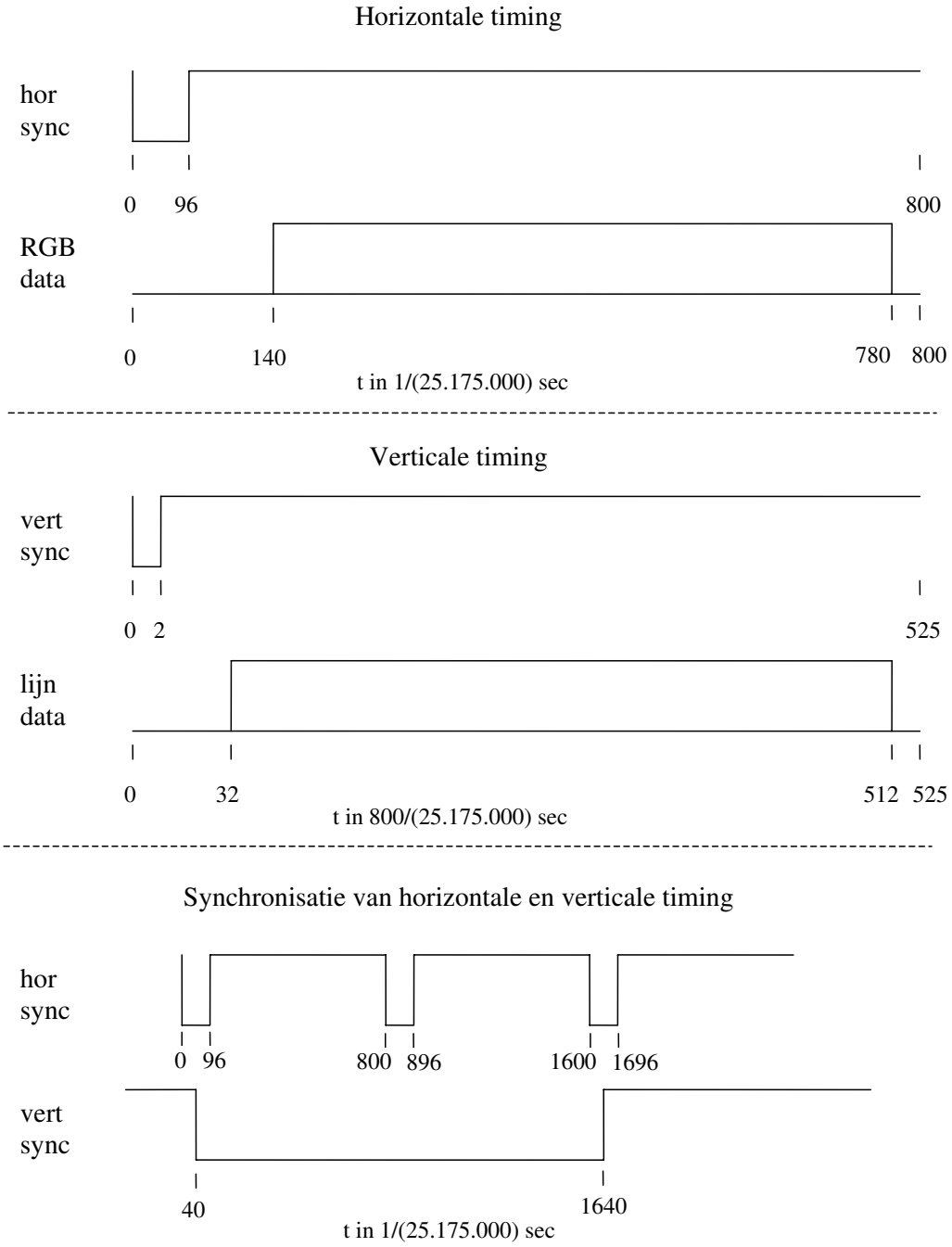
8.6.3 De spelbesturing

Elke speler heeft twee drukknopen: één om het racket omhoog te laten gaan en één om het racket omlaag te laten gaan. Verder zijn er:

- een reset knop om de stand op 0 - 0 te zetten en het spel te starten.
- een schakelaar waarmee 2 verschillende balsnelheden kunnen worden ingesteld:
  - horizontaal: ongeveer 500 pixels/sec, verticaal: ongeveer 250 pixels/sec.
  - horizontaal: ongeveer 1000 pixels/sec, verticaal: ongeveer 500 pixels/sec.
- een schakelaar waarmee 2 verschillende afmetingen van de rackets kunnen worden ingesteld:
  - 64 pixels verticaal
  - 32 pixels verticaal

8.6.4 De aansturing van de monitor

De chip dient een VGA-signaal te genereren voor een beeld van 640x480 pixels. Een korte schematische beschrijving van deze standaard is te zien in figuur 28. Voor verdere informatie, zie bijvoorbeeld http://www.epanorama.net/documents/pc/vga_timing.html. Merk op dat 5 uitgangssignalen nodig zijn voor een VGA-signaal: De waarde voor het rode pixel, de waarde voor het groene pixel, de waarde voor het blauwe pixel, een horizontaal sync-signaal en een verticaal sync-signaal. De laatste twee signalen zijn digitale signalen van 0 of 5 Volt en corresponderen dus met de op de Sea-Of-Gates chip gebruikte waarden voor een logische 0 en 1. De RGB pixelwaarden kunnen in principe analoge waarden hebben tussen 0 en 0.7 Volt. Voor ons spel zullen we de RGB pixels alleen helemaal aan of helemaal uitzetten en zullen er 3 externe level-shift schakelingen beschikbaar zijn waarme de 5 Volt uitgangswaarde van de Sea-Of-Gates chip kan worden omgezet naar 0.7 Volt. Hoewel volgens de definitie van de 640x480 pixel mode VGA-standaard een klok van 25.175 MHz vereist is (om elke klokpuls van 39.7 nsec. een R, G en B pixelwaarde over sturen), blijkt deze frequentiewaarde niet zo kritisch te zijn en zullen de meeste VGA-monitors nog werken wanneer er een afwijking van 5 % in de timing optreedt. Merk verder op dat wanneer de resolutie van de objecten in het spel niet te groot wordt gekozen en elke reeks van n (n = 2, 4 of 8) achtereenvolgende pixels dezelfde waarde hebben er verder ook volstaan kan worden met een n keer zo lage klokfrequentie.



Figuur 28: VGA-timing

8.6.5 Het ”pong” geluid

Via een kleine luidspreker van 50 Ohm die direct op de chip wordt aangesloten dient het ”pong” geluid gegenereerd te worden (een 1kHz blokgolf signaal).

8.6.6 De overige randvoorwaarden

Voor dit ontwerp mag een chip-oppervlak van 2 bond_bars worden gebruikt.

8.7 Alternatieve opdracht

Naast bovengenoemde opdrachten kan een groep ook zelf een opdracht bedenken en zelf de specificaties opstellen. Enige suggesties voor een alternatieve opdracht zijn:

- Een ander soort spelcomputer: bijvoorbeeld Snake, Tetris, Packman, etc.
- Een thermostaat.

Zo’n soort opdracht dient (1) aan dezelfde algemene randvoorden te voldoen als de andere opdrachten (zie sectie ’Algemene Randvoorwaarden’ hieronder), en (2) eenzelfde complexiteit te hebben (d.w.z. een chip-oppervlak van 1 à 2 bond_bars te omvatten). Verder dient de opdracht eerst door de begeleiding te worden goedgekeurd voordat met de verdere uitwerking wordt begonnen.

8.8 Algemene randvoorwaarden

- Voor de FSM’s (Finite State Machine) mogen alleen die van het Moore-type gebruikt worden. Meer informatie hierover is te vinden in [2].
- Als de schakeling geactiveerd wordt moeten alle FSM’s in hun begintoestand komen door middel van een reset signaal.
- Er moet bij het ontwerpen rekening gehouden worden met het feit dat de schakeling later getest zal worden. Als minimum eis wordt gesteld dat alle grote blokken apart getest moeten kunnen worden en dat van de verschillende besturingen ook de huidige toestand gelezen moet kunnen worden.
- Voor de opwekking van het kloksignaal kan gebruik gemaakt worden van een kristal van 6.144 MHz of 32 kHz.
- Het streven is om zo weinig mogelijk componenten extern te gebruiken. De dissipatie van de chip dient echter ook beperkt te zijn. Dit geeft een compromis voor de maximale stroom die de elektronica mag dissiperen voor de aansturing van de LEDs, etc.
- De voedingsspanning van het IC bedraagt 5 Volt.
- Het IC wordt gemaakt in een semi-custom CMOS proces waarvan de specificaties in appendix B vermeld zijn.

- Het beschikbare chip-oppervlak is circa 0.4 cm^2, hetgeen overeenkomt met ongeveer 40.000 transistor-paren, die gelijkelijk zijn verdeeld over 2 bond_bars (zie bibliotheek cellen)
- Er zijn op het IC, naast de voedingspinnen, per bond_bar 32 aansluitingen beschikbaar. Hoewel een totale schakeling meer dan één bond_bar aan oppervlakte kan hebben, kunnen bij de afmontage slechts de aansluitingen van één bond_bar worden aangesloten. In uitzonderlijke gevallen is het mogelijk om het aantal aansluitingen te verhogen d.m.v. multiplextechnieken.

8.9 Aanpak systeemspecificatie

In de middagen waarin de systeemspecificatie moet worden vastgesteld kunnen een aantal taken worden onderscheiden die één voor één doorlopen moeten worden om uiteindelijk tot een eenduidige systeemspecificatie te komen. Hieronder zijn deze taken opgesomd.

- Randvoorwaarden vaststellen vanuit de opdrachtbeschrijving (kan thuis worden voorbereid).
- Toevoegen van externe randvoorwaarden, complementeren van randvoorwaarden en specificaties.
- Korte beschrijving in woorden maken wat de schakeling moet doen.
- Verdeling van de schakeling in afzonderlijke deelsystemen.
- Beschrijving in woorden wat de verschillende deelsystemen moeten doen.
- Beschrijving van de interface (communicatie tussen de verschillende deelsystemen).
- Globale uitwerking van de deelsystemen. Vaststellen van de deelsysteemspecificaties en randvoorwaarden. Denk hierbij ook aan de testbaarheid.
- Onderlinge afstemming van de deelsystemen (naamgeving, timing, signalen, frequenties). Vaststellen welke signalen naar buiten worden gevoerd (ook voor testdoeleinden). Zorg ervoor dat de verdeling zo is dat de deelsystemen apart testbaar zijn.
- Verdeling van de deelsystemen over de groep, maken van een tijdsplanning. Afspraken maken voor onderling overleg.

8.10 Beschikbare ruimte

Per groep is er op de chip een ruimte beschikbaar die gevormd wordt door 2 bond_bars onder elkaar. Het ontwerp kan bestaan uit 1 geheel, waarbij er slechts 1 bond_bar is aangesloten en er dus 1 manier van afmonteren zal zijn, of het ontwerp kan bestaan uit 2 afzonderlijke gedeelten die elk apart aan een bond_bar zijn aangesloten en er dus 2 manieren van afmonteren zullen zijn. Wanneer er gekozen wordt voor 1 geheel zal deze schakeling als geheel moeten kunnen functioneren, maar zal deze schakeling het ook moeten toelaten om de afzonderlijke deelschakelingen te kunnen testen. Wanneer er gekozen wordt voor 2 gedeelten zal normaal gesproken het eerste deel als gehele schakeling functioneren en zullen op het twee deel de afzonderlijke deelschakelingen nog eens aanwezig zijn om apart getest te kunnen worden.

8.11 Testbaar ontwerpen

Nadat de schakelingen gefabriceerd zijn zullen zij door de groep getest worden. Het testen van de ontworpen schakeling na fabricage heeft een tweeledig doel:

- Nagaan of de schakeling aan de specificaties voldoet.
- Indien de schakeling niet aan de specificaties voldoet: proberen zo nauwkeurig mogelijk te weten te komen wat de reden is waardoor de schakeling niet voldoet, zodat tijdens een eventueel herontwerp de schakeling wel zal kunnen werken.

Het is van groot belang dat al in een vroeg stadium tijdens het ontwerp rekening wordt gehouden met het feit dat de schakeling later getest zal worden. Dit omdat in de eindfase van het ontwerp het meestal lastig is om de schakeling hiervoor nog aan te passen. Daarom wordt in deze sectie besproken hoe het testen zal plaatsvinden en aan welke eisen de schakeling (en de simulaties) moeten voldoen om later op goede wijze te kunnen testen.

8.11.1 De manier van testen bij het OP

Het testen van de schakeling na fabricage wordt op de volgende 2 manieren uitgevoerd. In beide gevallen wordt gemeten aan 2 a 3 stuks afgemonteerde schakelingen in een DIL40 behuizing.

- De schakeling wordt aangesloten op een logic analyzer. Via de logic analyzer worden logische signalen naar de ingangen van de schakeling gestuurd en de logische signalen aan de uitgangen worden geobserveerd. Dit is vergelijkbaar met het simuleren van de schakeling tijdens het ontwerp, maar nu met de echte hardware i.p.v. het (VHDL) simulatie model. Om te kunnen beoordelen of de schakeling werkt worden testvectors van een eerdere VHDL simulatie gebruikt en wordt gekeken of de uitgangen van de werkelijke schakeling dezelfde resultaten geven als tijdens simulatie (aannemende dat de eerder VHDL simulatie goede resultaten gaf). Voor meer informatie over deze vorm van testen, zie appendix D van de OP handleiding. De gehele schakeling dient op deze manier te worden getest, maar ook de deelschakelingen dienen op deze manier te worden getest.
- Aan de schakeling worden de externe componenten aangesloten (schakelaars, oscillator kristal voor de klok, displays, luidsprekers etc) en de schakeling wordt real-time uitgeprobeerd.

8.11.2 Overwegingen tijdens het ontwerp

Vanwege de testbaarheid van het ontwerp en het eventuele localiseren van problemen, dient aan de volgende punten aandacht te worden geschonken tijdens het ontwerp:

- Zorg dat de uitgangen van de schakeling, bij een gegeven set van ingangsignalen en na de reset, altijd een bekende, vaste, reeks van waarden gaan doorlopen. Alleen dan namelijk kan de logic analyzer een vergelijking maken met de referentie set van uitgangsignalen zoals eerder verkregen via simulatie.

- Tijdens het testen met de logic analyzer (en ook tijdens simulatie !) kan slechts over een beperkt aantal klokcycli gesimuleerd worden: enkele miljoenen/honderdduizenden cycli tijdens simulatie en 65000 cycli tijdens testen met de logic analyzer. Dat wil dus zeggen dat wanneer een kloksignaal van 6.144 Mhz gebruikt wordt en men het gedrag van de schakeling na 1 minuut (real-time) wil bekijken er 60 x 6144000 is ongeveer 360 miljoen klokcycli nodig zouden zijn, wat dus veel te veel is. Om dit probleem op te lossen kan men bijvoorbeeld oplossingen bedenken in de volgende richtingen: (1) de schakeling een bepaalde mode geven waarin hij met een veel langzamere klok toch de gewenste toestanden doorloopt of (2) het mogelijk maken dat de schakeling van een bepaalde begintoestand start waarna de schakeling veel sneller de gewenste eindtoestand bereikt.
- Wanneer een schakeling niet goed werkt is aan alleen de uitgangssignalen vaak niet altijd te zien waar het probleem zit. Probeer daarom zoveel mogelijk "interessante" interne signalen op de pinnen van het IC aan te sluiten wanneer er nog pinnen beschikbaar zijn. Slecht een beperkt aantal signalen kan tijdens het testen bereikt worden worden: naast voedingsaansluitingen zijn er slechts 32 pinnen beschikbaar per schakeling voor in en uitgangen. Om te analyzeren wat er wel werkt en niet werkt en waar het probleem dus zit kan hier gedacht worden aan het demultiplexen en multiplexen van in en uitgangssignalen zodat via 1 pin meerdere ingangssignalen worden aangeboden of uitgangen worden gelezen. Dit kost echter weer wel extra pinnen voor het aansturen van de (de)multiplexers.

8.11.3 Overwegingen tijdens het maken van testbenches

Voor zowel de gehele schakeling als de verschillende deelschakelingen zal minstens één van de testbenches die gebruikt is tijdens simulatie, tevens (na conversie) gebruikt worden voor tijdens het testen met de logic analyzer. Deze testbench zal aan de volgende eisen moeten voldoen:

- Gebruik een simulatie die niet langer duurt dan 65000 klokcycli.
- Een ingangsignaal moet gedurende de hele simulatie ook ingangsignaal blijven. D.w.z. het is niet toegestaan dat voor bepaalde tijd een logische waarde opdrukt aan een pin en de rest van de simulatie wordt het signaal "losgelaten".
- Bij de gespecificeerde ingangsignalen moeten na afloop van de reset altijd de gespecificeerde uitgangsignalen optreden (de uitgangsignalen mogen na de reset dus niet afhankelijk zijn van een toevallige willekeurige begintoestand waarin de schakeling zich bevindt).

8.12 Afronding ontwerp

Wanneer de uiteindelijke layout gemaakt gaat worden dient allereerst aan de totale schakeling de bibliotheekcel osc10 aangesloten te worden om het kloksignaal aan te sturen. De aansturing van het kloksignaal dient plaats te vinden via terminal F van de osc10 cel. De terminals XI en XO dienen als aansluitpinnen voor het kristal naar buiten te worden gevoerd (dus aangesloten aan de bond_bar) en de terminal E kan aan VDD worden vastgemaakt. Van het geheel dient nu een layout gemaakt te worden zodanig dat deze past op 1 of 2 bond_bars (afhankelijk van hoe gekozen wordt de beschikbare ruimte op de chip te gebruiken). Het is vaak handig om deze laatste plaatsing van de deelschakelingen

met de hand te doen waarbij 1 of 2 bond_bars op de achtergrond worden geplaatst als referentiekader. Vergeet dan echter niet om de bond_bars na afloop weer te verwijderen.

De samenvoeging van de totale ontworpen layout met een bond_bar dient te gebeuren met het commando Add bond_bar uit het Utility menu van *GoWithTheFlow*. Hierbij wordt gespecificeerd welke terminals uit het ontwerp vast zitten aan welke IC pinnen. Er wordt hierbij een nieuwe circuit cel gegenereerd bestaande uit 2 componenten (de hierboven ontworpen layout en de bond_bar) die dan vervolgens met Place & route moet worden omgezet in een layout. Plaats de ontworpen component hierbij met de hand over de bond_bar en route ze aan elkaar met *trout* zonder de optie border terminals te gebruiken.

Om de kans op een werkende schakeling te vergroten is het verder **belangrijk dat deze uiteindelijke layout (inclusief bond_bar) altijd als laatste nog eens geverifieerd wordt door er een switch-level simulatie voor uit te voeren**. Deze switch-level simulatie kan bijvoorbeeld gebaseerd zijn op de testbench die later ook gebruikt zal worden voor het testen met de logic analyzer.

Maak tenslotte van de totale layout, inclusief bond_bar, een ldm file met het commando Make ldm.

De files die dan bij de beleiding moeten worden ingeleverd zijn:

- De LDM file voor de totale layout, aangemaakt met het commando Make ldm
- De bijbehorende .buf file met daarin de informatie over hoe de pinnen zijn aangesloten. (Deze file wordt automatisch gegenereerd tijdens het uitvoeren van Utilities → Add bond_bar.)
- Een .ref file voor elke test die na fabricage verricht zal gaan worden voor de schakeling. (Een .ref file wordt aangemaakt uit een .lst file met behulp van het commando Generate → Reference file.) Het is gewenst om een voor elke deelschakeling een .ref file te hebben en een .ref voor de totale schakeling.

A Enige aanwijzingen voor het gebruik van Linux

A.1 Eerste maal inloggen

Voor het practicum moet worden ingelogd onder linux, gebruikmakende van je NetID gebruikersnaam en password. Voor problemen hiermee kun je terecht bij het EWI steunpunt bij de portiersloge in het gebouw aan de Drebbelweg.

Na het inloggen moet een terminal window worden worden geopend en moet daarin het volgende commando worden getypt:

```
source /data/public/common/software/opprog/bin/op_init
```

Dit zorgt ervoor, dat alle paden naar de te gebruiken programmatuur worden gezet. Alle programma's die tijdens het practicum gebruikt worden zijn nu bereikbaar. Deze procedure behoeft slechts eenmalig bij de eerste keer inloggen te worden uitgevoerd.

A.2 Enkele linux commando's

Enkele veelgebruikte linux commando's die in een terminal window gegeven kunnen worden zijn:

ls Toon de inhoud van een de huidige directory. De optie -l geeft details over elke file en subdirectory.

mkdir Maak een nieuwe directory.

mkdir <dir_name>	maak een een directory <dir_name> onder de huidige directory.

cd Verander van directory.

cd	ga naar de home_directory
cd ..	ga naar de parent_directory
cd <dir_name>	ga naar de opgegeven directory

cp Copieer een file.

cp <file_from> <file_to>	maak een copie van <file_from> onder de naam <file_to>
cp <file_from> <dir_to>	maak een copie van <file_from> in directory <dir_to> met de naam <file_from>

rm Verwijder een file.

rm <file_name>	verwijder de file <file_name> uit het file_systeem.
rm -rf <dir_name>	verwijder de directory <dir_name> en al zijn files en subdirectories uit het file_systeem

man Geef uitleg over een commando.

man rm	geef uitleg over het commando rm

Voor file en directory namen kunnen pattern matching karakters zoals het karakter * gebruikt worden. Bijvoorbeeld, het volgende commando copieert alle files die eindigen op .vhd naar een directory proj/VHDL

```
cp *.vhd proj/VHDL
```

B De Sea-of-Gates Chip

B.1 Globale beschrijving

De chip waar de OP practicum-ontwerpen op gemaakt worden is van het ”semi-custom”type. Dit houdt in dat de grootte van de chip vast staat en dat ook de grootte, ligging en aantal van de te gebruiken transistoren al bekend is. In feite is de chip al voor een groot deel vervaardigd, alleen de metallisatie die nodig is voor de gewenste schakeling, moet nog worden aangebracht. De gekozen structuur op de reeds gefabriceerde wafers (plakken silicium met een doorsnede van 4 inch) is die van de moderne gate-array: de Sea-of-Gates structuur. Deze structuur leent zich primair voor digitale schakelingen, maar analoge zijn ook mogelijk.

Fabricageproces:

Een 1.6 μm nwell-CMOS proces (het Philips C3DM proces).

Grootte van de chip:

- 10 x 10 mm, (46 exemplaren op een 4 inch plak)
- 144 aansluitpinnen waarvan de functie programmeerbaar is door middel van metallisatiepatronen,
- 200.000 transistoren (de helft is pmos, de andere helft is nmos),
- afmeting nmos transistor: 1.6 x 23.2 μm,
- afmeting pmos transistor: 1.6 x 29.6 μm.

De metallisatie van de wafers, het uitzagen en afmonteren van de chips gebeurt in het DIMES instituut. Voor de metallisatie zijn 4 maskerstappen nodig: 2 metaal- en 2 kontaktgaten maskers.

De grootte van de chip is zodanig gekozen dat meerdere typen ontwerpen gemaakt kunnen worden. Ook is het niet bij voorbaat onmogelijk om een redelijk grote schakeling te realiseren. De belangrijkste voorwaarde is dat DIMES in staat moet zijn de metallisatie aan te brengen. Hieronder volgt een lijst met de maskers die DIMES voor de metallisatie moet maken (en dus door de ontwerper gespecificeerd) en hun functie. Voor de volledigheid is een opsomming toegevoegd van de maskers die nodig waren om de sea-of-gates chip te maken.

DIMES maskers:

- co (con, cop, cps) : Definieert posities van kontaktgaten in het oxide tussen de onderste metaallaag(in) en od en ps gebieden.
- in : Definieert de onderste (eerste) metaallaag.
- cos : Definieert de positie van kontaktgaten in het oxide tussen eerste en tweede(bovenste) metaallaag(ins).

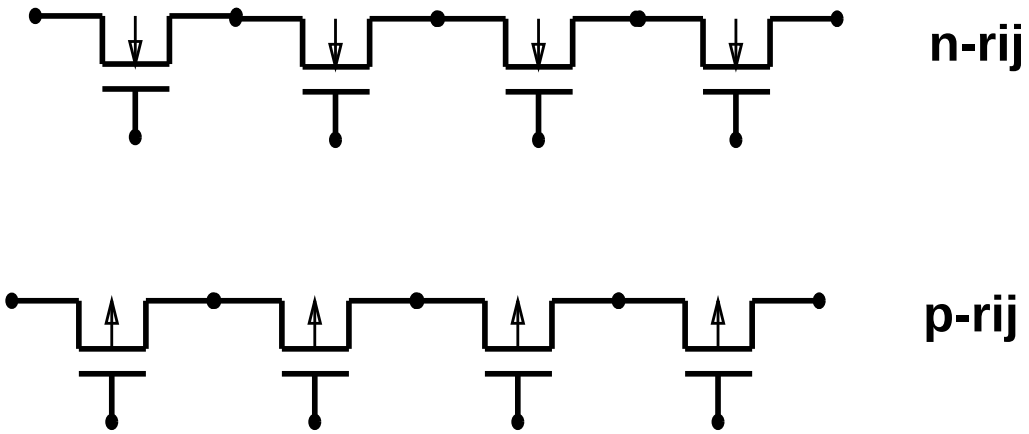
- ins : Definieert de bovenste (tweede) metaallaag.

Overige maskers:

- nw : nwell, definieert de gebieden waarin p-transistoren kunnen worden gedefinieerd.
- od : Definieert de actieve gebieden: plaats van source, gate en drain van beide typen transistoren.
- sn : Definieert die delen van de actieve gebieden die van het n-type zijn.
- sp : Definieert die delen van de actieve gebieden die van het p-type zijn.
- ps : Definieert gates van transistoren.

B.2 De kern van de chip, vanuit circuit-standpunt

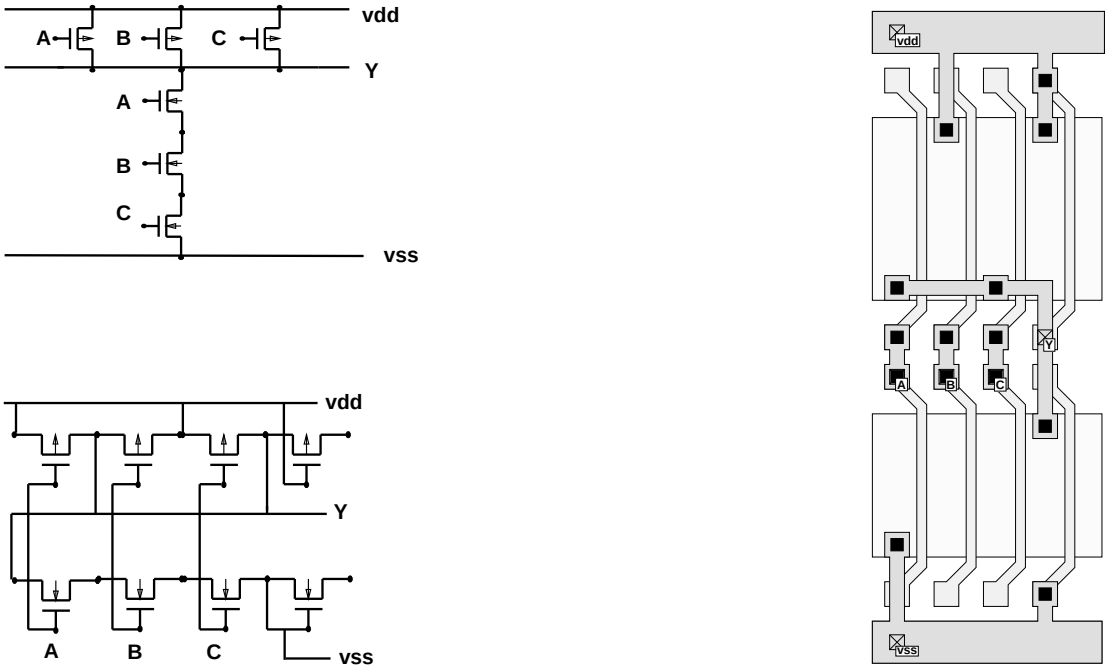
De chip heeft een sea-of-gates structuur en bestaat eenvoudig uit een groot aantal rijen transistoren zoals hieronder getekend. Er zijn evenveel rijen n- als p-type transistoren (zie figuur 29)



Figuur 29: Schematische voorstelling van een pmos- en een nmos-rij op de sea-of-gates chip

Elke rij op de chip telt ongeveer 1000 transistoren. Elk knooppunt in elke rij is toegankelijk vanuit de eerste metaallaag. De tweede metaallaag kan gebruikt worden indien kruisingen tussen metaalsporen nodig zijn. Naast een rij n-transistoren ligt altijd een rij p-transistoren.
De rijenstructuur lijkt onhandig maar is dit in de praktijk niet. In een schakeling zijn namelijk altijd rijen transistoren te ontdekken.
Als voorbeeld een 3-input nand schakeling (zie figuur 30).

In de structuur is duidelijk dat alle transistoren keurig op een rij gelegd kunnen worden en met een eenvoudig verbindingspatroon (in metaal) verbonden kunnen worden tot een 3-input nand. De in de structuur meest rechts getekende transistoren dienen alleen als scheiding van de schakeling met de rest van de transistoren. De herhaling in de y-richting van n- en p-rijen transistoren is als volgt: `....nppnnppnnppn...` (zie ook figuur 31).



Figuur 30: Schema van de 3-input nand, de afbeelding hiervan op een rijenstructuur en de layout op het sea-of-gates image

Dit heeft o.a. als reden dat de p-transistoren op deze manier meer gegroepeerd zijn voor grotere nwell gebieden.
De basisrepetitie van 4 rijen (`nppn`) komt op de chip 44 keer voor.
In de tabellen 1 en 2 staan enkele elektrische eigenschappen van het cmos proces.

masker	weerstand ($\Omega/\square$)
n^+ OD	55
p^+ OD	75
poly-Si	25-60
in	0.045
ins	0.03

Tabel 1: Nominale interconnectie weerstand cmos proces

B.3 De kern van de chip, vanuit een layout-standpunt

In figuur 31 is een stukje layout gegeven van de basisstructuur. Duidelijk zijn de rijen transistoren zichtbaar. De p-transistor is breder dan de n-transistor. Het verschil in breedte is gelijk aan de ruimte die nodig is voor een extra metaalspoor en kontaktgat op de actieve gebieden van de p-transistoren. Tussen de twee rijen p-transistoren is plaats voor het voedingsspoor met daaronder een od gebied

	Oppervlakte capaciteit (aF/μm^2)	Rand capaciteit (aF/μm)	opmerking
n^+ OD naar substraat	190	310	junctie capaciteit
p^+ OD naar substraat	450	570	junctie capaciteit
poly naar substraat	49	54	
metaal1 naar substraat	25	45	
metaal 1 naar OD	49	54	
metaal1 naar poly	49	55	
metaal2 naar substraat	13	49	
metaal2 naar OD	15	53	
metaal2 naar poly	22	62	
metaal1 naar metaal2	43	81	

Tabel 2: Nominale interconnectie capaciteiten cmos proces

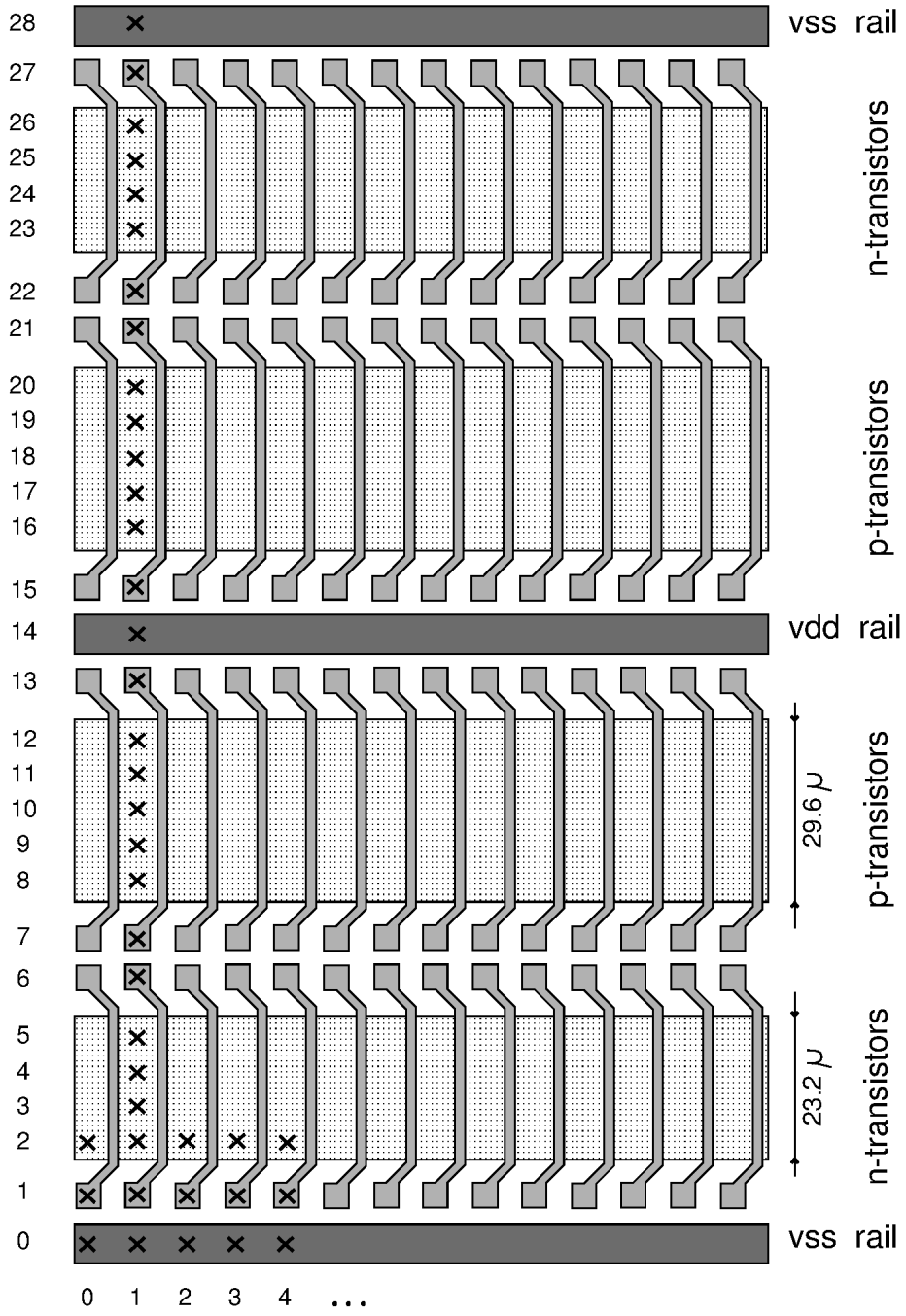
om de nwell te kunnen verbinden met de hoogste spanning in de schakeling (de voedingsspanning). Onder het aardspoor ligt een od gebied om het p-substraat aan aarde te kunnen leggen.

Ook valt nog op dat alle afmetingen rechthoekig zijn, behalve de gate-aansluitingen waarin 45-graden lijnen zijn gebruikt. Dit is gedaan uit efficiency overwegingen.
Wat de layout betreft:

De steek in de x-richting is 8.0 μm.
De steek in de y-richting is 197.6 μm.
De layout-eenheid is 1 lambda (1 lambda = 0.2 μm).

B.4 Eisen en afspraken met betrekking tot de metallisatie

- Horizontaal lopend over het actieve gebied van een rij n-transistoren is plaats voor maximaal 4 metaalsporen.
  Horizontaal lopend over het actieve gebied van een rij p-transistoren is plaats voor maximaal 5 metaalsporen.
- Source- en drain gebieden en ook gate-aansluitingen) kunnen alleen maar verbonden worden met metaal1 (in). Verbinding van source, drain en gate met metaal2 (ins) kunnen alleen maar lopen via metaal1.
- Kontakten van metaal1 met source, drain of gate wordt gemaakt middels kontaktgaten. Deze kontaktgaten mogen alleen gedefinieerd worden op de grid-punten (zie figuur 31). Kontakt van metaal1 (in) en pmos source en draingebied wordt gemaakt met cop, kontakt van metaal1 (in) en nmos source en draingebied wordt gemaakt met con en kontakt van metaal1 (in) met de gates (poly-silicium) wordt gemaakt met cps.
  Overigens mogen deze drie kontaktlagen (con, cop en cps) bij het maken van de layout door elkaar worden gebruikt omdat bij het maken van het masker voor deze gaten ook geen onderscheid wordt gemaakt.



Figuur 31: Layout van het fishbone sea-of-gates image.

- Kontakten tussen metaal1 (in) en metaal2 (ins) worden gemaakt met cos.
- Een con- of cop- of cps-kontakt en een cos kontakt mogen niet boven elkaar worden gedefinieerd.
- Op de plaats van een gate kontaktaansluiting mag geen cos kontakt worden gedefinieerd.
- Metaal1 (in) loopt overwegend horizontaal, metaal2 (ins) loopt overwegend verticaal.

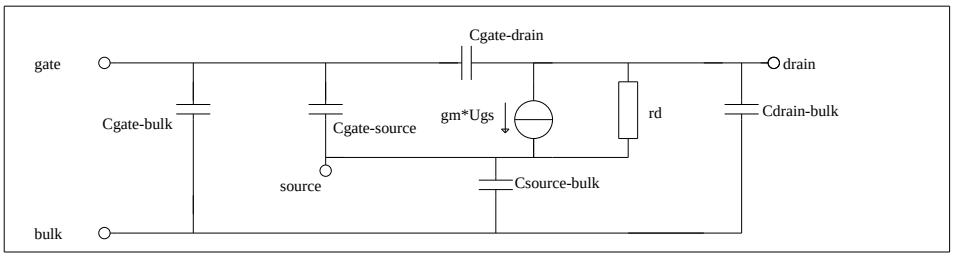
B.5 Meetgegevens van de Sea-of-Gates chip

Karakteristieken van de nmos en pmos transistoren In figuur 33 is de gemeten drainstroom I_d uitgezet tegen de drain-sourcespanning V_{ds} met de gate-sourcespanning V_{gs} als parameter.
In figuur 34 is de gemeten drainstroom I_d uitgezet tegen de gatespanning V_{gs} met de drain-sourcespanning V_{ds} als parameter.
De maximale stroom voor een enkele nmos transistor (V_{gs} = V_{ds} = 5 V) is ca. 5.5 mA. Voor een enkele pmos transistor is de maximale stroom (V_{gs} = V_{ds}= -5 V) gelijk aan ca. 2.7 mA.

Kleinsignaalmodel van de mos transistor In figuur 32 is een eenvoudig kleinsignaalmodel gegeven van de mos transistor. De waarde van de verschillende capaciteiten zijn nog niet gemeten.
Voor de drainstroom in verzadiging geldt ($V_{ds} > V_{gs} > V_t$, zie ook [3]):

$$I_d = \frac{\beta}{2}(V_{gs} - V_t)^2(1 + \lambda V_{ds})$$

De gemeten waarde van de verschillende parameters zijn te vinden in tabel 3.



Figuur 32: Een eenvoudig kleinsignaalmodel van de mos transistor

Parameter	nmos	pmos
V_t	0.7 V	-1.2 V
β	0.65 mA/V^2	0.13 mA/V^2
λ	0.027 V^{-1}	0.096 V^{-1}
r_d	7.5 kΩ	10 kΩ

Tabel 3: Enkele gemeten parameters van de nmos en pmos transistoren

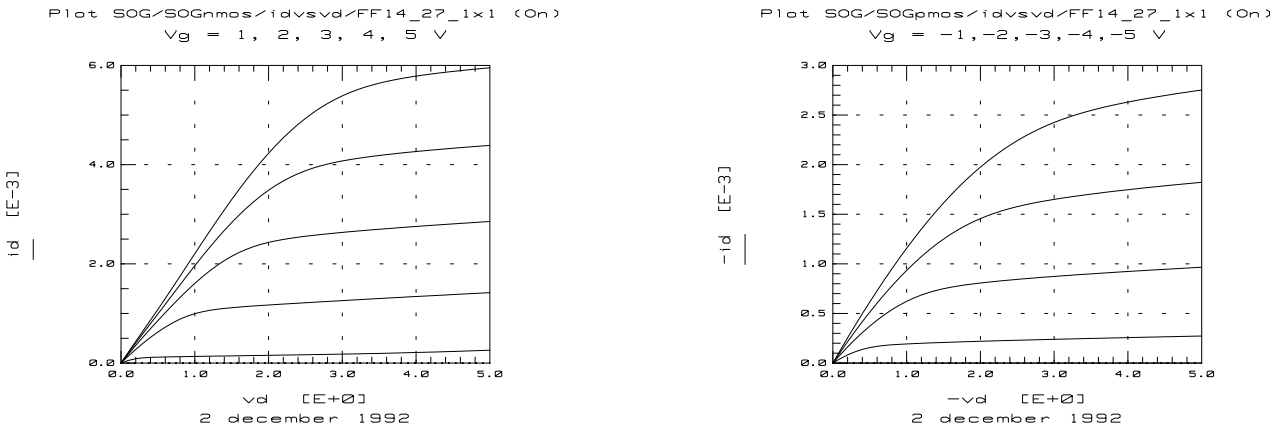
Weerstandswaarden In tabel 4 zijn de gemeten waarden te vinden van een aantal weerstanden en vergeleken met de berekende waarden uit gegevens van het cmos-proces.

Grootheid	Berekende waarde	Gemeten waarde
Contactgateweerstand		
in-ins (cos)	max. 0.2 Ω	0.13 Ω
in-ps (cps)	max. 50 Ω	22 Ω
Metaalweerstand		
metaal1 (**in**)	45 mΩ/□	56 mΩ/□
metaal2 (**ins**)	30 mΩ/□	26 mΩ/□
Polysiliciumgateweerstand		
nmos gate	690 Ω	700 Ω
pmos gate	790 Ω	950 Ω

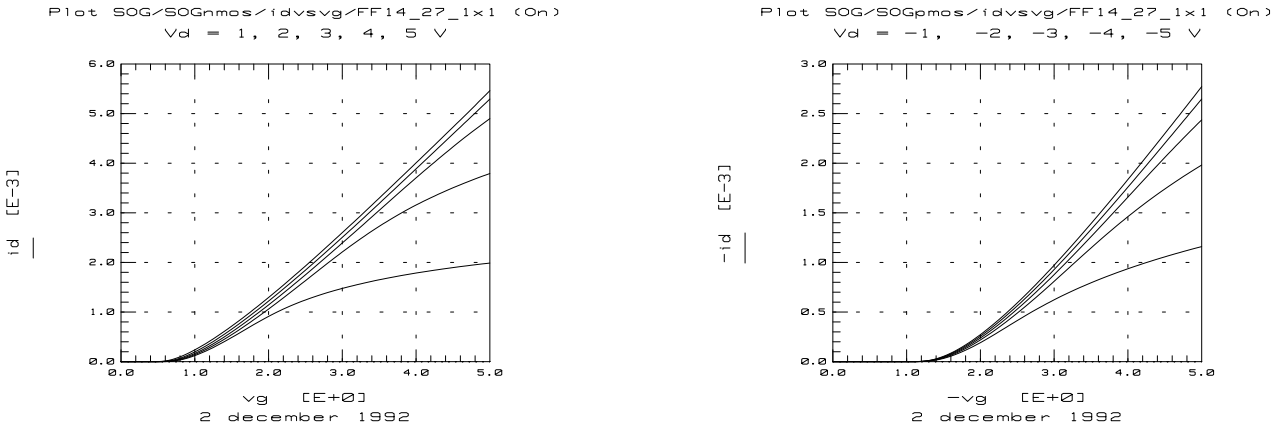
Tabel 4: Enkele weerstandwaarden van de sea-of-gates chip

B.6 Structuur in de rand van de chip

Er zijn in de rand 144 aansluitposities, verdeeld in 8 groepen van elk 18. Elke groep van 18 aansluitingen heeft een voedingspositie en een aardpositie. De andere 16 zijn programmeerbaar d.m.v. metallisatiepatronen tot een aantal bufferfuncties, b.v.: digitaal in, digitaal uit, analoog in, analoog uit. Alle 8 aardposities in de rand zijn met elkaar verbonden (via de chiprand). Voor de programmering van de rand van een chip wordt verwezen naar een afzonderlijke handleiding. Voor de liefhebbers liggen er in de ruimten tussen de buffers in de rand nog polyweerstanden. In elke tussenruimte liggen er 8 van elk 2.5 kΩ (= 100 vierkanten). Tenslotte ligt in de rand linksonder een grote poly-rechthoek die nuttig kan zijn voor een capaciteit. Deze weerstanden en capaciteiten zijn echter niet te gebruiken in het ontwerppracticum omdat ze vanwege de kwarten-indeling van de chip niet zijn aan te sluiten.



Figuur 33: De gemeten drainstroom van een nmos en pmos transistor uitgezet tegen de drain-sourcespanning met de gate-sourcespanning als parameter



Figuur 34: De gemeten drainstroom van een nmos en pmos transistor uitgezet tegen de gate-sourcespanning met de drain-sourcespanning als parameter

C De OP Cellenbibliotheek (oplib)

C.1 Inleiding

In dit hoofdstuk zal een korte beschrijving gegeven worden van de bibliotheekcellen die beschikbaar zijn voor het ontwerppracticum (OP). De beschrijving bestaat voor elke cel uit:

- Functie
- Terminal aansluitingen
- IEC-symbool
- Waarheidstabel
- Timing parameters
- Equivalent chip oppervlak
- Fanout

Er zijn vier soorten timing parameters:

T_{PLH} en T_{PHL}	De vertragingstijd van de cel bij eenheidsbelasting (0.12 pF)
ΔT_{PLH} en ΔT_{PHL}	De vertragings-coëfficiënt bij capacitieve belasting
C_{in}	De ingangscapaciteit
T_{su} en T_{hold}	Setup- en hold-tijden van de flipflops

De vertragings-tijden en -coëfficiënten worden afzonderlijk gedefinieerd voor een opgaande (LH) en neergaande (HL) flank van het uitgangssignaal. De totale vertragingstijd wordt berekend met de formule:

$$T_{P_{tot}} = T_P + \Delta T_P(C_{load} - C_{unit})$$

waarbij C_{load} de belastingscapaciteit voorstelt en C_{unit} de eenheidsbelasting.
Deze belastingscapaciteit is de som van de ingangscapaciteit van de aangestuurde cellen plus de capaciteit van de verbindingsdraden.

De eenheid van (equivalent) chip oppervlakte is gedefinieerd als het kleinst mogelijke stukje van het fishbone image en bestaat uit één nmos en één pmos transistor. De grootte hiervan is ongeveer 800 μm^2.

De fanout is de belasting waarboven de totale vertragingstijd *afneemt* als een buffer wordt toegevoegd.

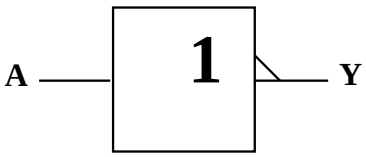
C.2 De cellen

C.2.1 iv110

Functie: Inverter

Terminals: (A, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	Y
L	H
H	L

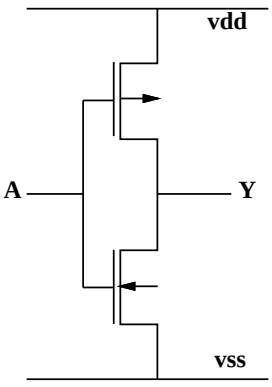
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.13	0.19	0.35	ns
T_{PHL}	A	Y	0.15	0.23	0.40	ns
ΔT_{PLH}	A	Y	-	-	1.1	ns/pF
ΔT_{PHL}	A	Y	-	-	0.8	ns/pF
C_{in}	A	vss	-	0.12	0.18	pF

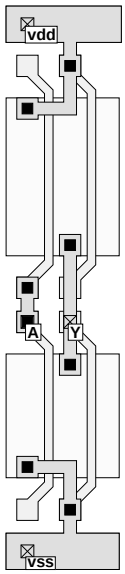
Equivalent chip oppervlak: 2

Fanout: 3.0 pF

Circuit:



Layout:

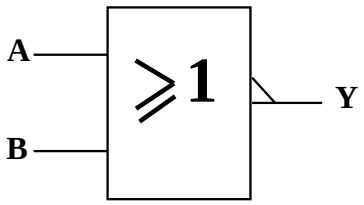


C.2.2 no210

Functie: 2-input nor

Terminals: (A, B, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	B	Y
L	L	H
-	H	L
H	-	L

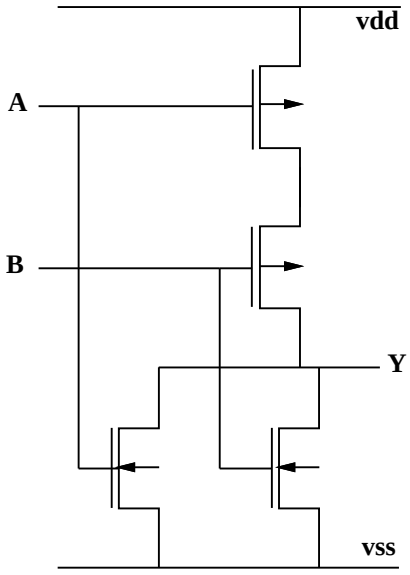
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.23	0.34	0.59	ns
T_{PHL}	A	Y	0.10	0.25	0.38	ns
T_{PLH}	B	Y	0.21	0.32	0.59	ns
T_{PHL}	B	Y	0.10	0.22	0.40	ns
ΔT_{PLH}	any	Y	-	-	1.8	ns/pF
ΔT_{PHL}	any	Y	-	-	0.8	ns/pF
C_{in}	any	vss	-	0.12	0.18	pF

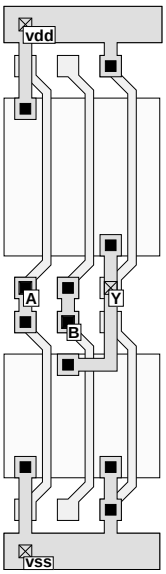
Equivalent chip oppervlak: 3

Fanout: 2.4 pF

Circuit:



Layout:

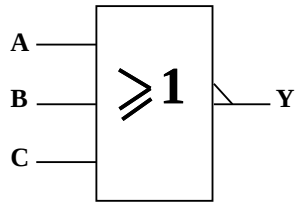


C.2.3 no310

Functie: 3-input nor

Terminals: (A, B, C, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	B	C	Y
L	L	L	H
-	-	H	L
-	H	-	L
H	-	-	L

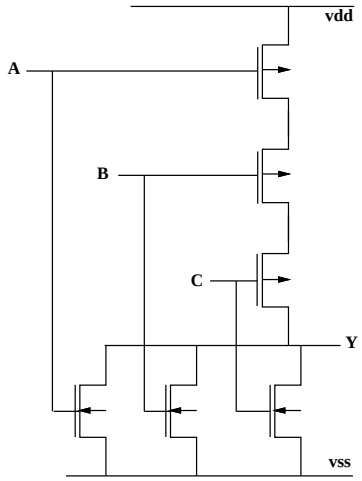
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.36	0.54	1.1	ns
T_{PHL}	A	Y	0.08	0.27	0.47	ns
T_{PLH}	B	Y	0.36	0.53	1.1	ns
T_{PHL}	B	Y	0.08	0.27	0.47	ns
T_{PLH}	C	Y	0.30	0.46	1.1	ns
T_{PHL}	C	Y	0.08	0.24	0.41	ns
ΔT_{PLH}	any	Y	-	-	2.4	ns/pF
ΔT_{PHL}	any	Y	-	-	0.9	ns/pF
C_{in}	any	vss	-	0.12	0.18	pF

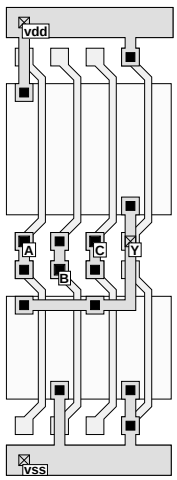
Equivalent chip oppervlak: 4

Fanout: 1.9 pF

Circuit:



Layout:

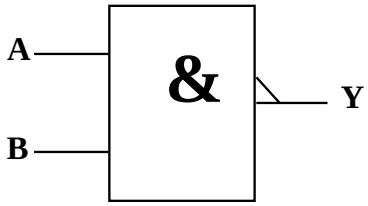


C.2.4 na210

Functie: 2-input nand

Terminals: (A, B, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	B	Y
-	L	H
L	-	H
H	H	L

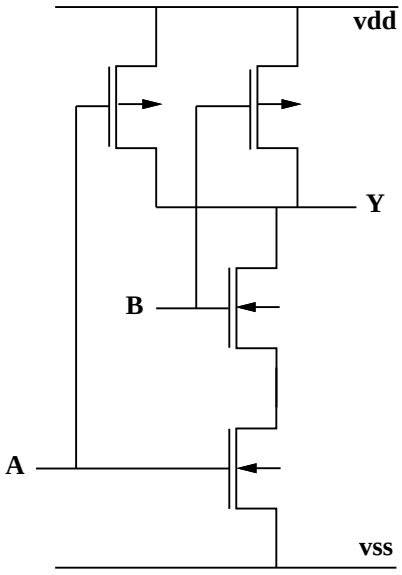
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.08	0.33	0.50	ns
T_{PHL}	A	Y	0.19	0.29	0.52	ns
T_{PLH}	B	Y	0.08	0.20	0.36	ns
T_{PHL}	B	Y	0.20	0.30	0.52	ns
ΔT_{PLH}	any	Y	-	-	1.0	ns/pF
ΔT_{PHL}	any	Y	-	-	1.1	ns/pF
C_{in}	any	vss	-	0.12	0.18	pF

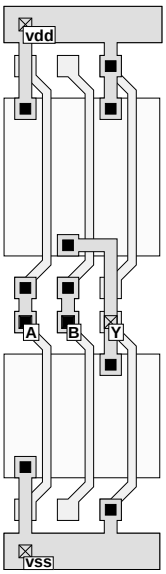
Equivalent chip oppervlak: 3

Fanout: 2.9 pF

Circuit:



Layout:

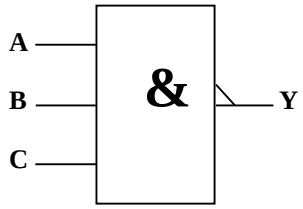


C.2.5 na310

Functie: 3-input nand

Terminals: (A, B, C, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	B	C	Y
-	-	L	H
-	L	-	H
L	-	-	H
H	H	H	L

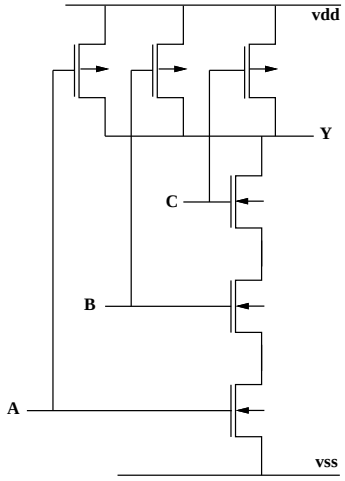
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.06	0.37	0.61	ns
T_{PHL}	A	Y	0.26	0.39	0.77	ns
T_{PLH}	B	Y	0.06	0.35	0.58	ns
T_{PHL}	B	Y	0.27	0.40	0.77	ns
T_{PLH}	C	Y	0.06	0.21	0.37	ns
T_{PHL}	C	Y	0.26	0.39	0.77	ns
ΔT_{PLH}	any	Y	-	-	0.9	ns/pF
ΔT_{PHL}	any	Y	-	-	1.4	ns/pF
C_{in}	any	vss	-	0.12	0.18	pF

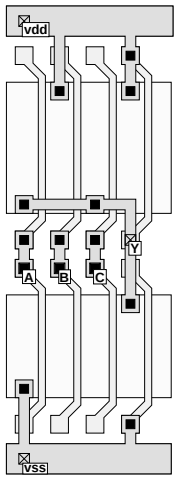
Equivalent chip oppervlak: 4

Fanout: 2.5 pF

Circuit:



Layout:

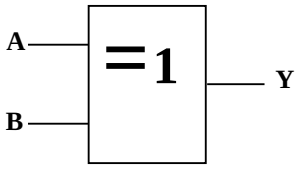


C.2.6 ex210

Functie: Exclusive or

Terminals: (A, B, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

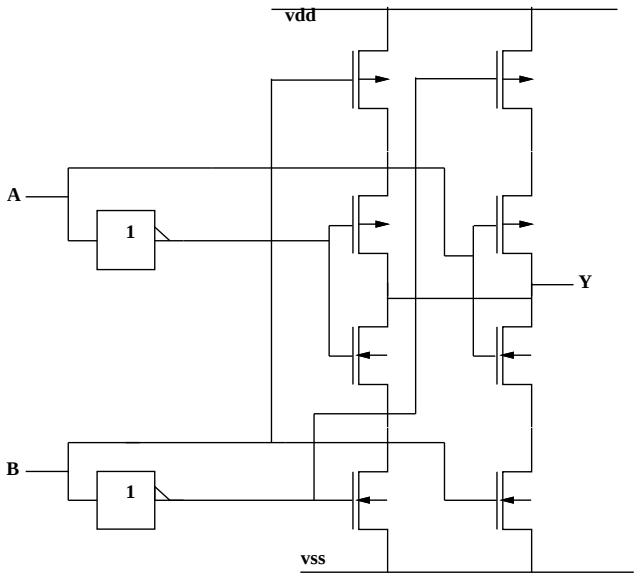
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	any	Y	0.22	0.69	1.6	ns
T_{PHL}	any	Y	0.24	0.41	0.68	ns
ΔT_{PLH}	any	Y	-	-	1.4	ns/pF
ΔT_{PHL}	any	Y	-	-	1.1	ns/pF
C_{in}	any	vss	-	0.24	0.36	pF

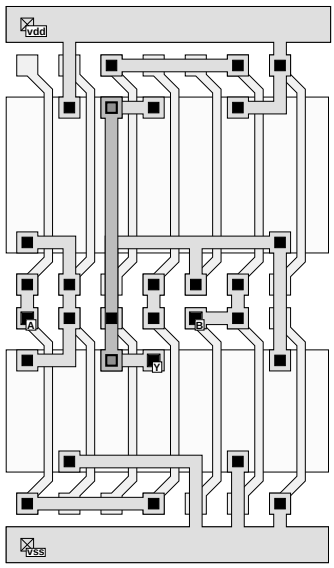
Equivalent chip oppervlak: 7

Fanout: 2.4 pF

Circuit:



Layout:

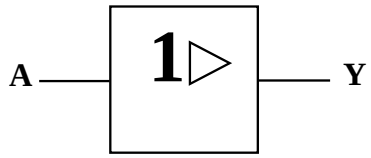


C.2.7 buf40

Functie: Buffer (4x-drive)

Terminals: (A, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

A	Y
L	L
H	H

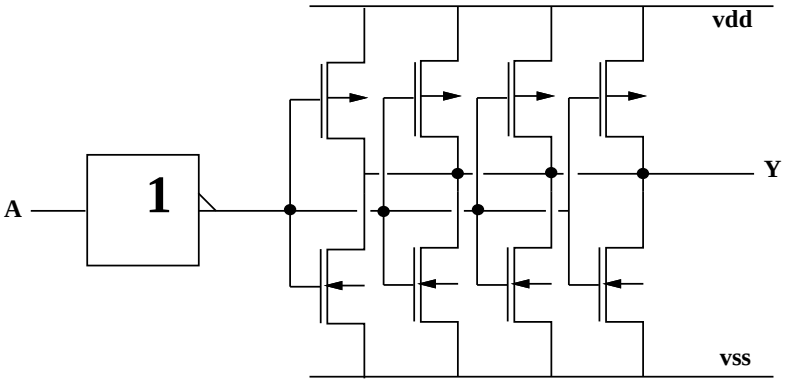
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y	0.46	0.69	1.0	ns
T_{PHL}	A	Y	0.56	0.84	1.3	ns
ΔT_{PLH}	A	Y	-	-	0.3	ns/pF
ΔT_{PHL}	A	Y	-	-	0.4	ns/pF
C_{in}	A	vss	-	0.12	0.18	pF

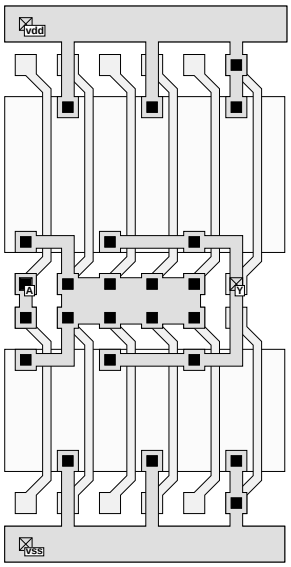
Equivalent chip oppervlak: 6

Fanout: 12.0 pF

Circuit:



Layout:

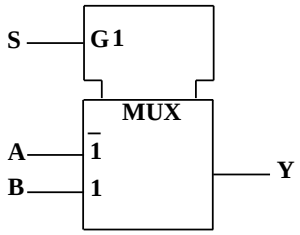


C.2.8 mu111

Functie: 2-line-to-1-line data selector/multiplexer

Terminals: (A, B, S, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

S	A	B	Y
L	L	-	L
L	H	-	H
H	-	L	L
H	-	H	H

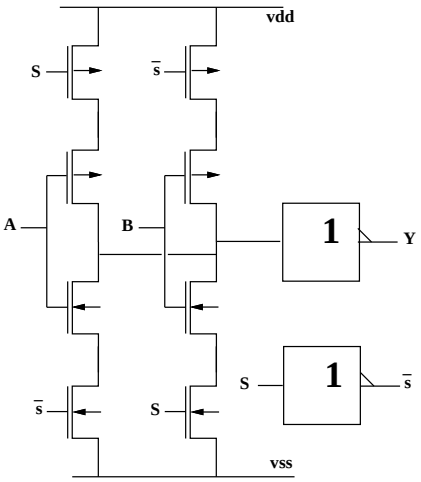
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A, B	Y	0.31	0.50	0.81	ns
T_{PHL}	A, B	Y	0.37	0.60	0.98	ns
T_{PLH}	S	Y	0.37	0.60	0.95	ns
T_{PHL}	S	Y	0.37	0.65	1.1	ns
ΔT_{PLH}	any	Y	-	-	1.2	ns/pF
ΔT_{PHL}	any	Y	-	-	1.0	ns/pF
C_{in}	A, B	vss	-	0.12	0.18	pF
C_{in}	S	vss	-	0.24	0.36	pF

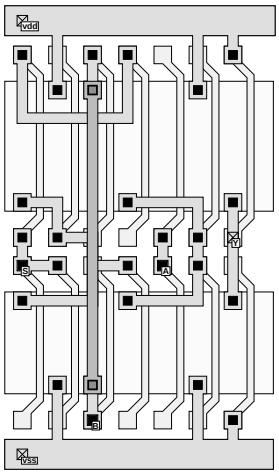
Equivalent chip oppervlak: 7

Fanout: 3.0 pF

Circuit:



Layout:

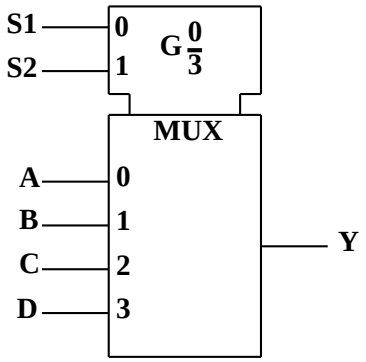


C.2.9 mu210

Functie: 4-line-to-1-line data selector/multiplexer

Terminals: (S1, S2, A, B, C, D, Y, vss, vdd)

IEC-symbool:



Waarheidstabel:

S2	S1	A	B	C	D	Y
L	L	L	-	-	-	L
L	L	H	-	-	-	H
L	H	-	L	-	-	L
L	H	-	H	-	-	H
H	L	-	-	L	-	L
H	L	-	-	H	-	H
H	H	-	-	-	L	L
H	H	-	-	-	H	H

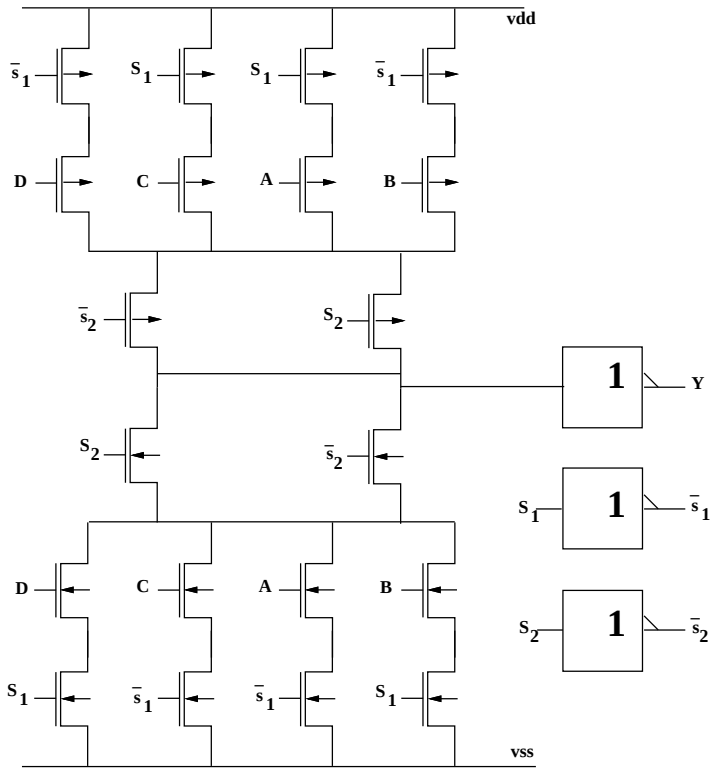
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A, B, C, D, S1	Y	0.88	1.3	2.0	ns
T_{PHL}	A, B, C, D, S1	Y	1.4	2.1	3.2	ns
T_{PLH}	S2	Y	0.77	1.2	1.7	ns
T_{PHL}	S2	Y	0.57	0.86	1.3	ns
ΔT_{PLH}	any	Y	-	-	1.3	ns/pF
ΔT_{PHL}	any	Y	-	-	1.4	ns/pF
C_{in}	A, B, C, D	vss	-	0.12	0.18	pF
C_{in}	S1	vss	-	0.36	0.54	pF
C_{in}	S2	vss	-	0.24	0.36	pF

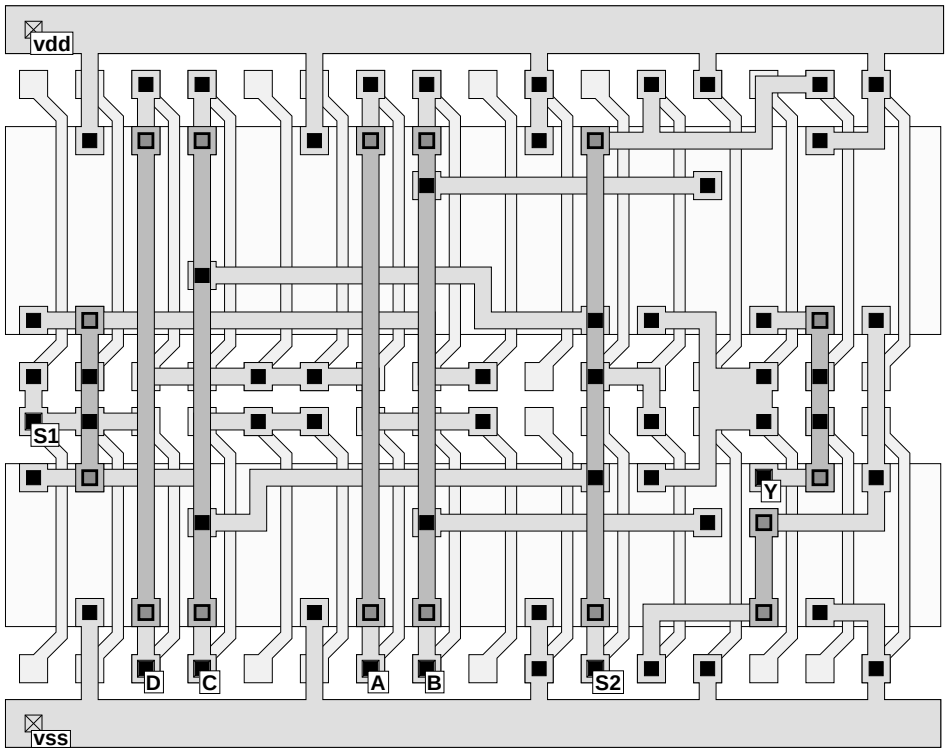
Equivalent chip oppervlak: 16

Fanout: 3.0 pF

Circuit:



Layout:

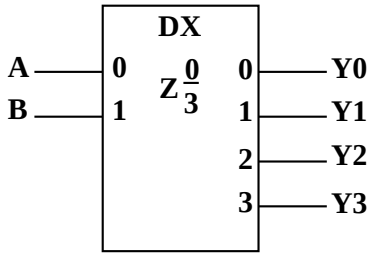


C.2.10 de211

Functie: 2-to-4 decoder/demultiplexer

Terminals: (A, B, Y0, Y1, Y2, Y3, vss, vdd)

IEC-symbool:



Waarheidstabel:

B	A	Y0	Y1	Y2	Y3
L	L	H	L	L	L
L	H	L	H	L	L
H	L	L	L	H	L
H	H	L	L	L	H

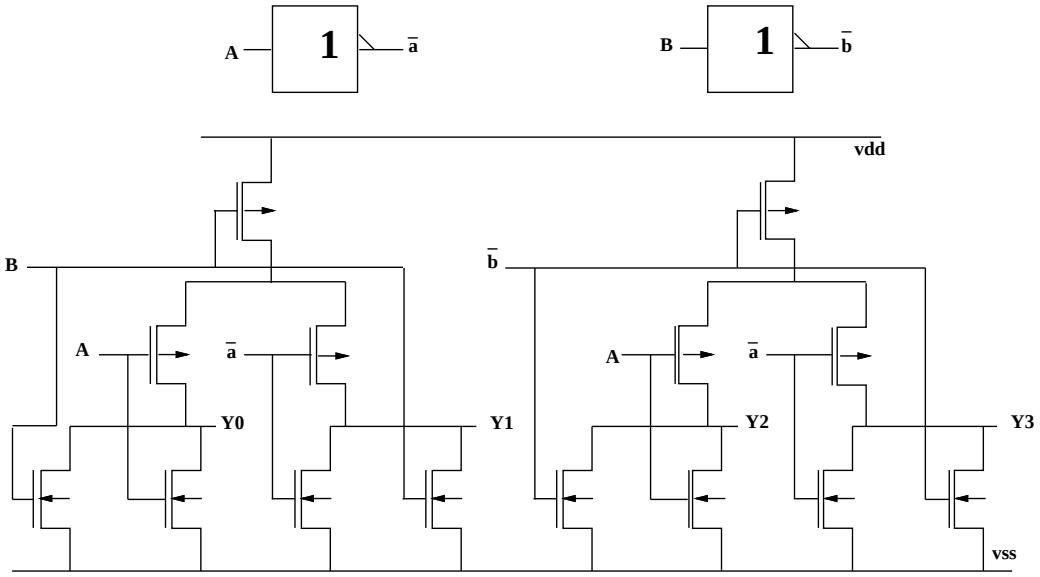
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	A	Y0,Y2	0.23	0.34	0.51	ns
T_{PHL}	A	Y0,Y2	0.17	0.25	0.38	ns
T_{PLH}	A	Y1,Y3	0.41	0.62	0.93	ns
T_{PHL}	A	Y1,Y3	0.36	0.54	0.81	ns
T_{PLH}	B	Y0,Y1	0.20	0.30	0.45	ns
T_{PHL}	B	Y0,Y1	0.15	0.22	0.33	ns
T_{PLH}	B	Y2,Y3	0.38	0.57	0.87	ns
T_{PHL}	B	Y2,Y3	0.34	0.51	0.76	ns
ΔT_{PLH}	any	any	-	-	1.8	ns/pF
ΔT_{PHL}	any	any	-	-	0.8	ns/pF
C_{in}	A	vss	-	0.36	0.54	pF
C_{in}	B	vss	-	0.30	0.45	pF

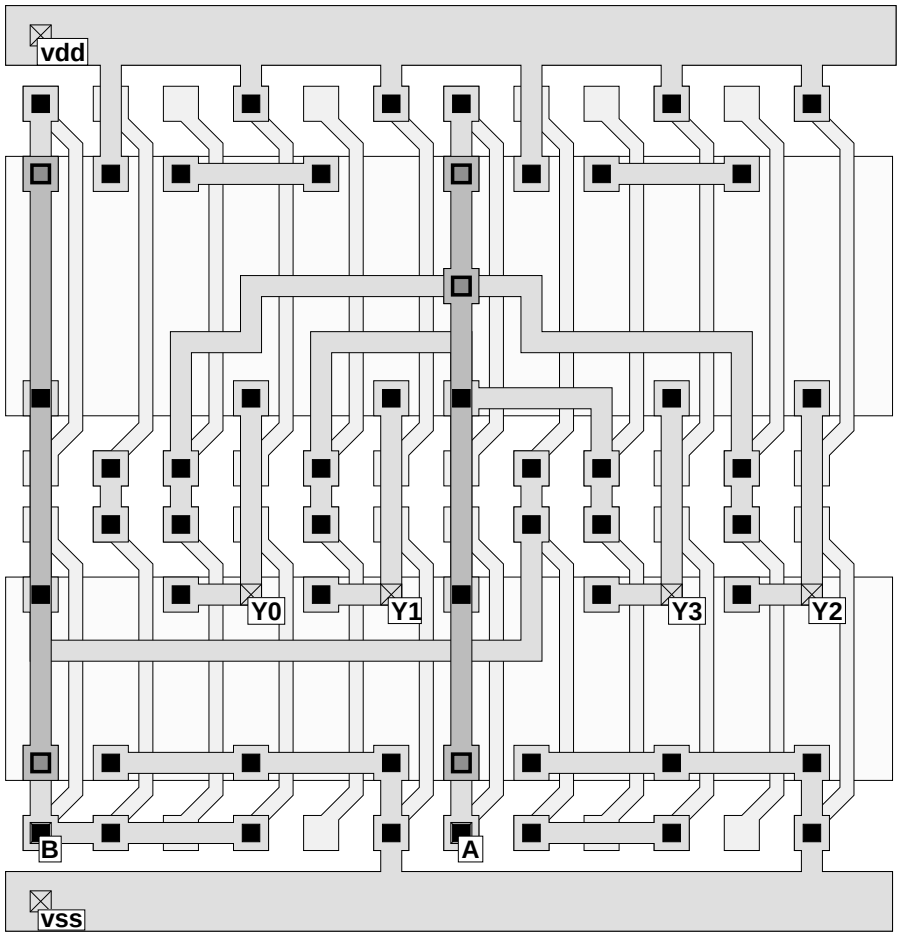
Equivalent chip oppervlak: 12

Fanout: 2.4 pF

Circuit:



Layout:

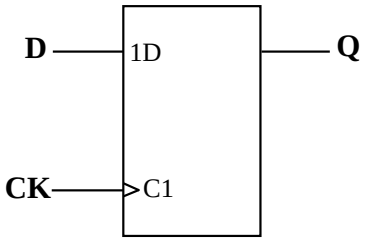


C.2.11 dfn10

Functie: D-type positive edge-triggered flipflop

Terminals: (D, CK, Q, vss, vdd)

IEC-symbool:



Waarheidstabel:

D	CK	Q
L	↑	L
H	↑	H

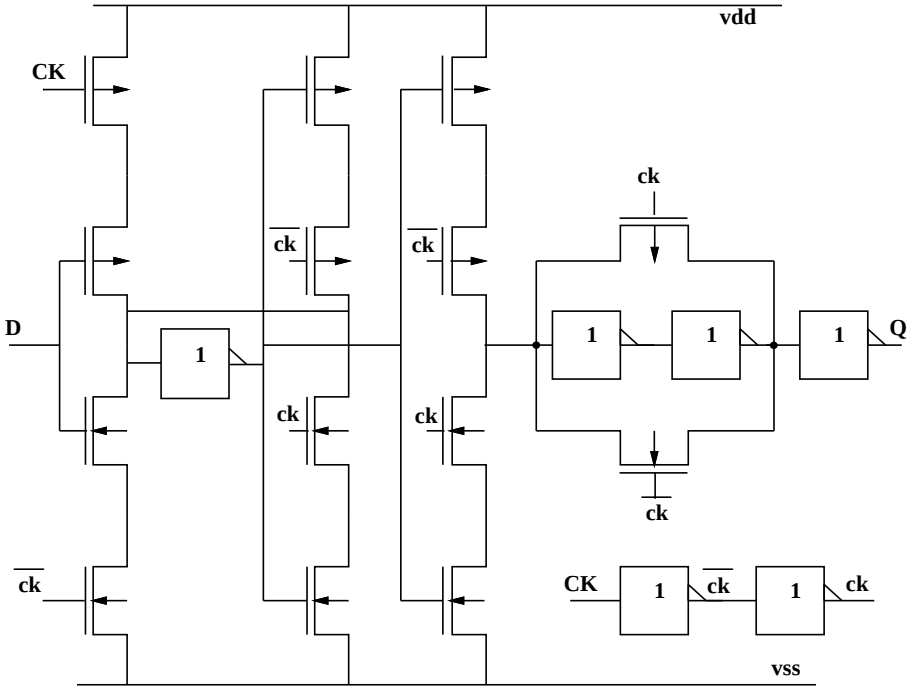
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	CK	Q	1.4	2.1	3.2	ns
T_{PHL}	CK	Q	1.3	1.9	2.9	ns
ΔT_{PLH}	CK	Q	-	-	1.1	ns/pF
ΔT_{PHL}	CK	Q	-	-	0.8	ns/pF
T_{su}	D	CK	-	-	1.4	ns
T_{hold}	D	CK	-	-	0.9	ns
C_{in}	D	vss	-	0.12	0.18	pF
C_{in}	CK	vss	-	0.18	0.27	pF

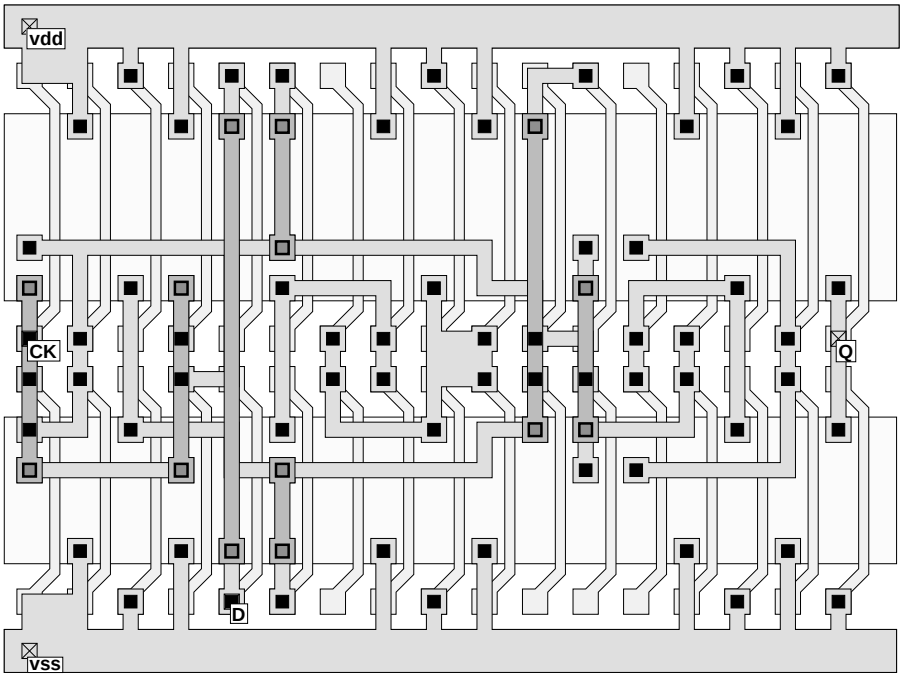
Equivalent chip oppervlak: 17

Fanout: 3.0 pF

Circuit:



Layout:

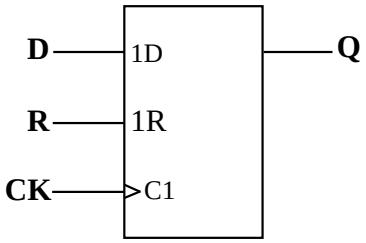


C.2.12 dfr11

Functie: D-type positive edge-triggered flipflop with synchronous reset

Terminals: (D, R, CK, Q, vss, vdd)

IEC-symbool:



Waarheidstabel:

R	D	CK	Q
L	L	↑	L
L	H	↑	H
H	-	↑	L

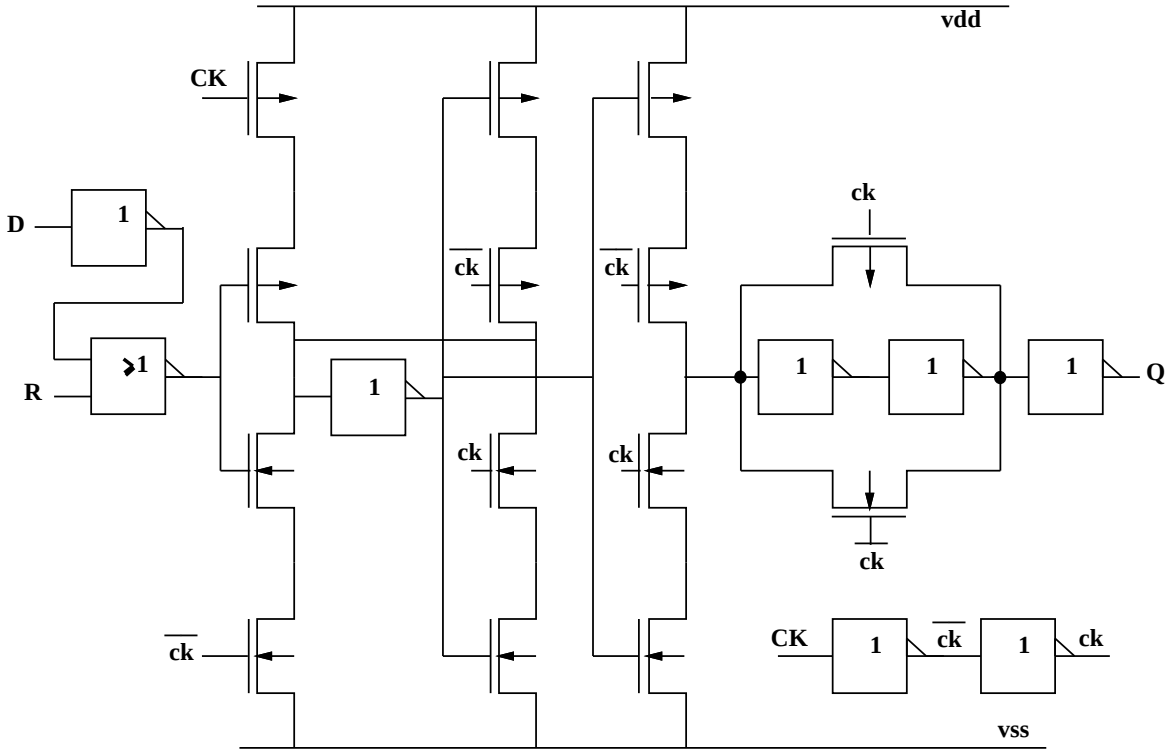
Timing parameters:

Parameter	From	To	Min	Typ	Max	Unit
T_{PLH}	CK	Q	1.4	2.1	3.2	ns
T_{PHL}	CK	Q	1.3	1.9	2.9	ns
ΔT_{PLH}	CK	Q	-	-	1.1	ns/pF
ΔT_{PHL}	CK	Q	-	-	0.8	ns/pF
T_{su}	D,R	CK	-	-	1.4	ns
T_{hold}	D,R	CK	-	-	0.9	ns
C_{in}	D,R	vss	-	0.12	0.18	pF
C_{in}	CK	vss	-	0.18	0.27	pF

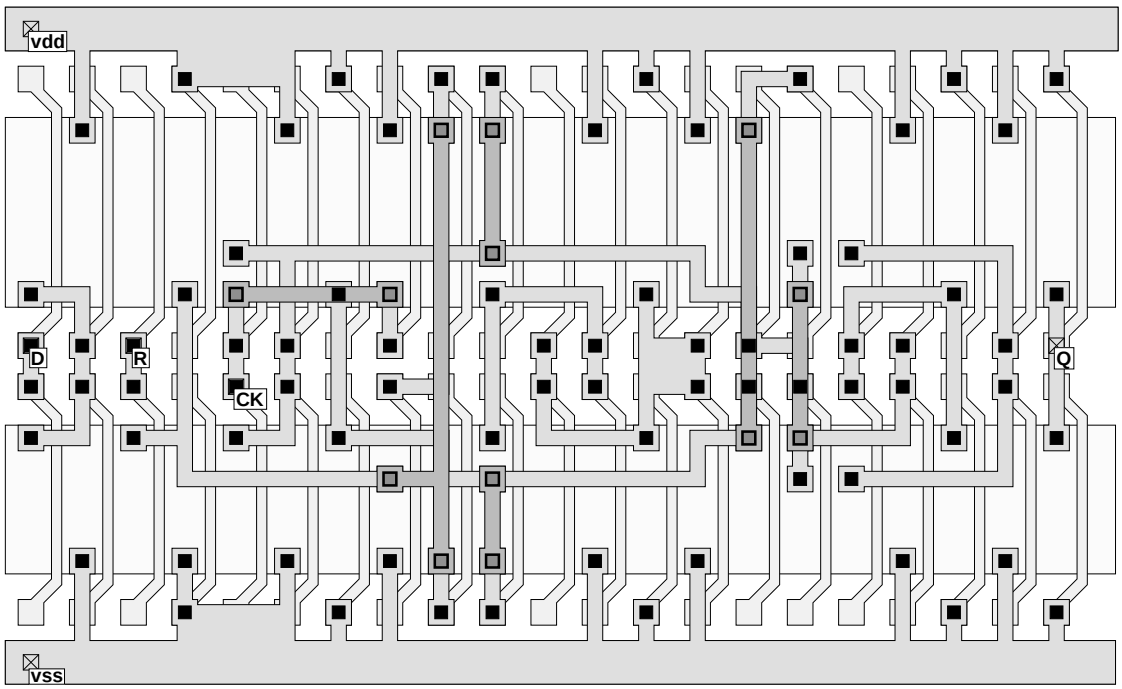
Equivalent chip oppervlak: 21

Fanout: 3.0 pF

Circuit:



Layout:

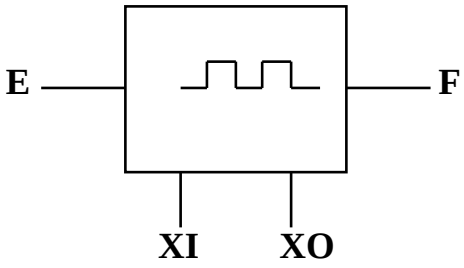


C.2.13 osc10

Functie: Crystal oscillator with enable

Terminals: (E, F, XI, XO, vss, vdd)

IEC-symbool:



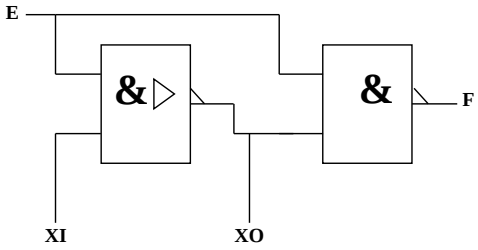
Frequentie bereik: 1 MHz t/m 20 MHz

Equivalent chip oppervlak: 16

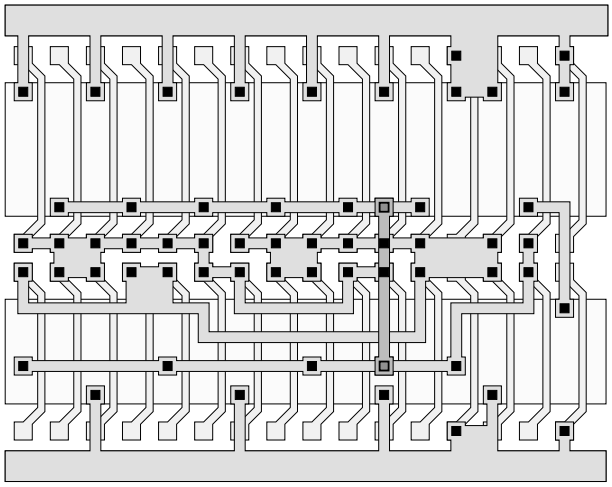
Beschrijving:

XI en XO zijn de aansluitingen voor het kristal. XI moet gebufferd worden met een inputbuffer, XO dient niet gebufferd te worden.
Als E (enable) hoog is, is de oscillator aktief. Als E laag is, is de uitgang F hoog.

Circuit:



Layout:



C.2.14 ln3x3

Functie: NMOS compoundtransistor, basiscel voor analoge schakelingen

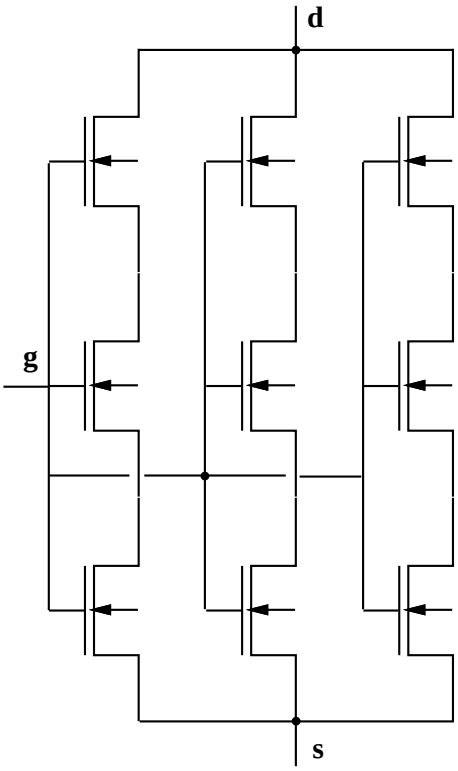
Terminals: (g, d, s, vss, vdd)

Equivalent chip oppervlak: 10

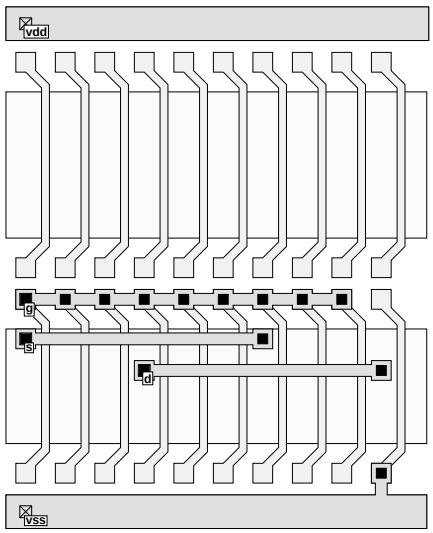
Beschrijving:

De NMOS compoundtransistor ln3x3 bestaat uit 9 NMOS basistransistoren die in een matrix van 3x3 geschakeld zijn.

Circuit:



Layout:



C.2.15 lp3x3

Functie: PMOS compoundtransistor, basiscel voor analoge schakelingen

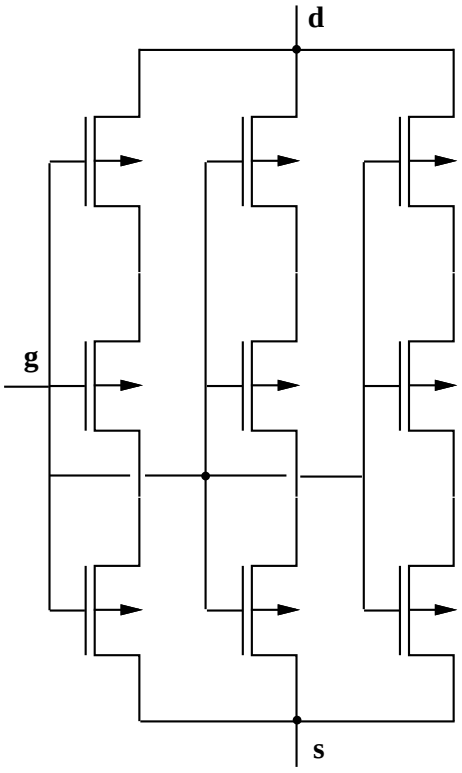
Terminals: (g, d, s, vss, vdd)

Equivalent chip oppervlak: 10

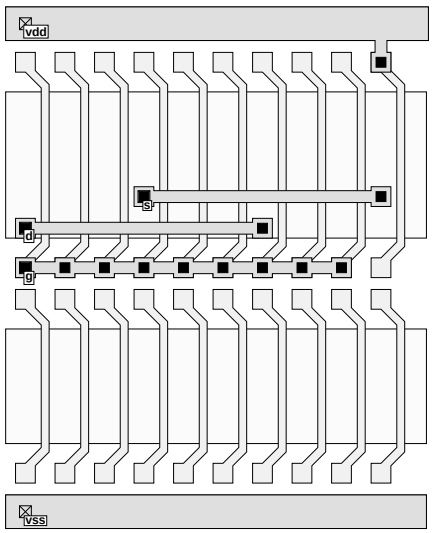
Beschrijving:

De PMOS compoundtransistor lp3x3 bestaat uit 9 PMOS basistransistoren die in een matrix van 3x3 geschakeld zijn.

Circuit:



Layout:



C.2.16 mir_nin, mir_nout

Functie: NMOS spiegel, basiscel voor stroomspiegel

Terminals mir_nin: (i, g, vss, vdd)

Terminals mir_nout: (i, g, o, vss, vdd)

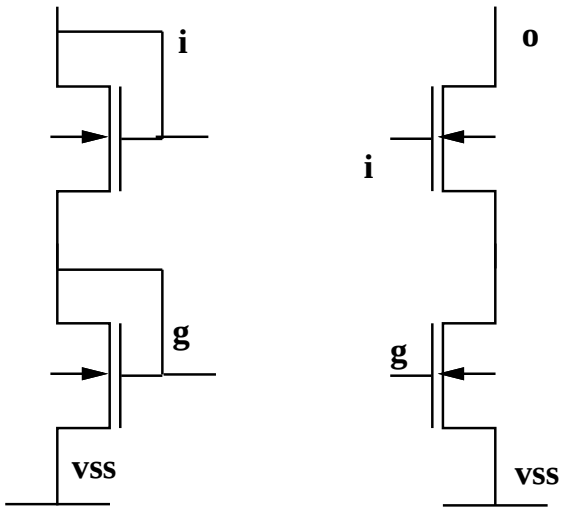
Equivalent chip oppervlak: 19

Beschrijving:

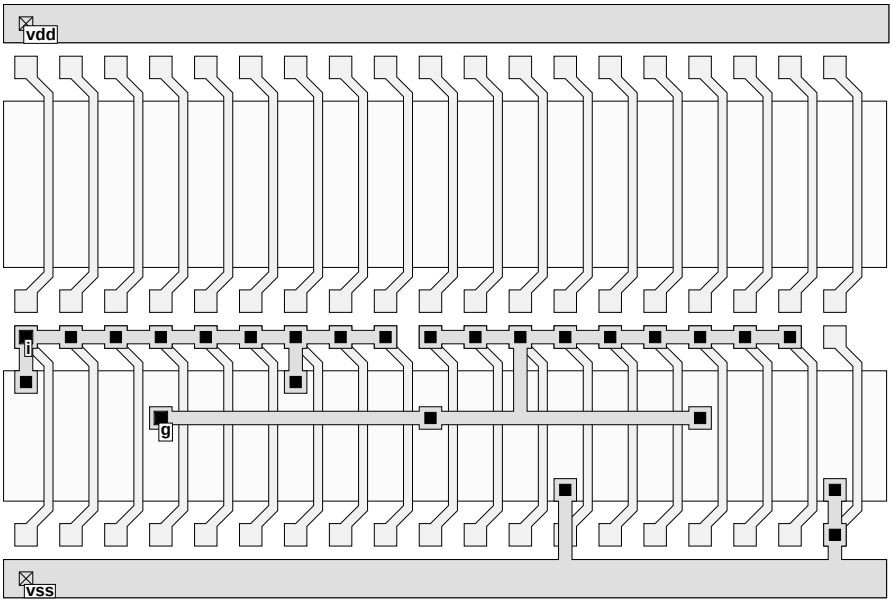
De NMOS spiegel is een gecascodeerde stroomspiegel. Hij bestaat uit een ingang mir_nin en een uitgang mir_nout. Beide kunnen in een spiegel meerdere keren gerepeteerd worden om schaling te verkrijgen. De gebruikte transistoren zijn zgn. compound transistoren.

Circuit:

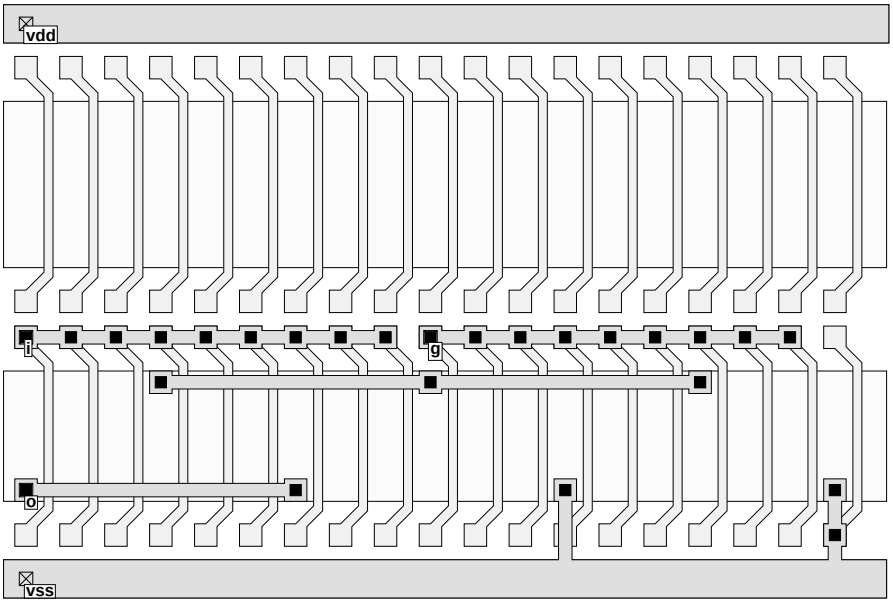
(a) mir_nin (b) mir_nout



Layout mir_nin:



Layout mir_nout:



C.2.17 mir_pin, mir_pout

Functie: PMOS spiegel, basiscel voor stroomspiegel

Terminals mir_pin: (i, g, vss, vdd)

Terminals mir_pout: (i, g, o, vss, vdd)

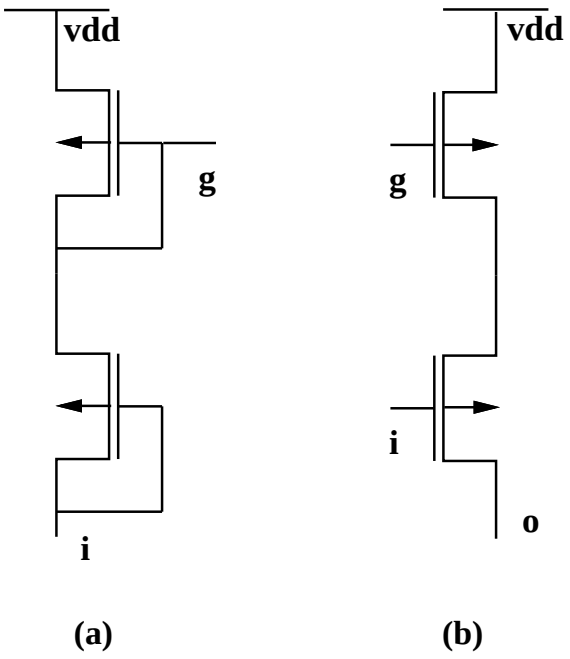
Equivalent chip oppervlak: 19

Beschrijving:

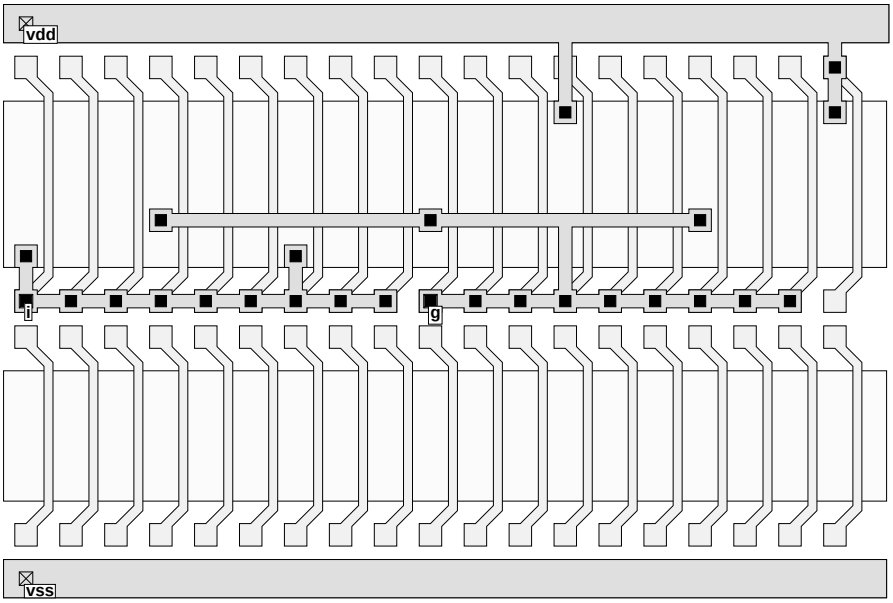
De PMOS spiegel is een gecascodeerde stroomspiegel. Hij bestaat uit een ingang mir_pin en een uitgang mir_pout. Beide kunnen in een spiegel meerdere keren gerepeteerd worden om schaling te verkrijgen. De gebruikte transistoren zijn zgn. compound transistoren.

Circuit:

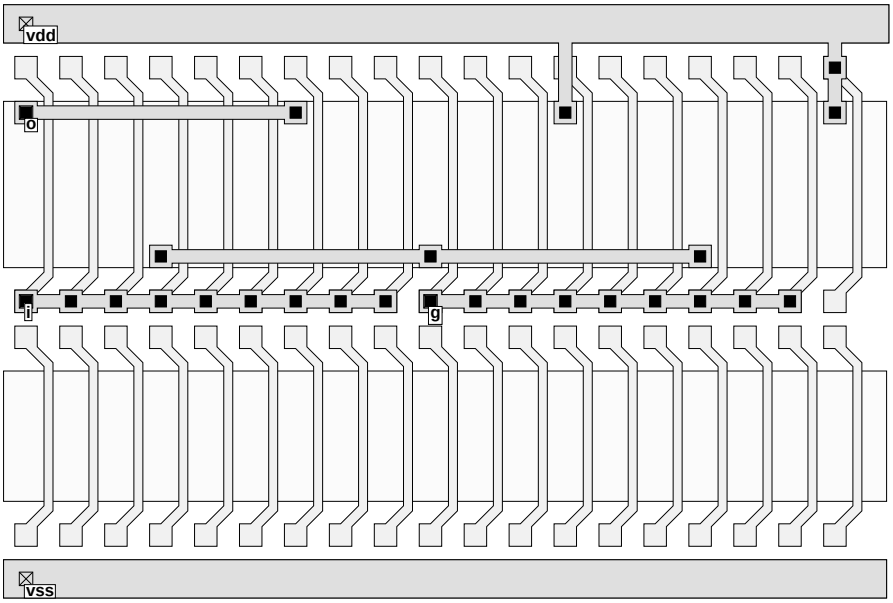
(a) mir_pin (b) mir_pout



Layout mir_pin:



Layout mir_pout:



C.2.18 bond_leer

Functie: Basiscel voor kleine SoG-schakelingen

Terminals: (vss, bf1, bf2, bf3, vdd, bf4, bf5, bf6)

Equivalent chip oppervlak: 800

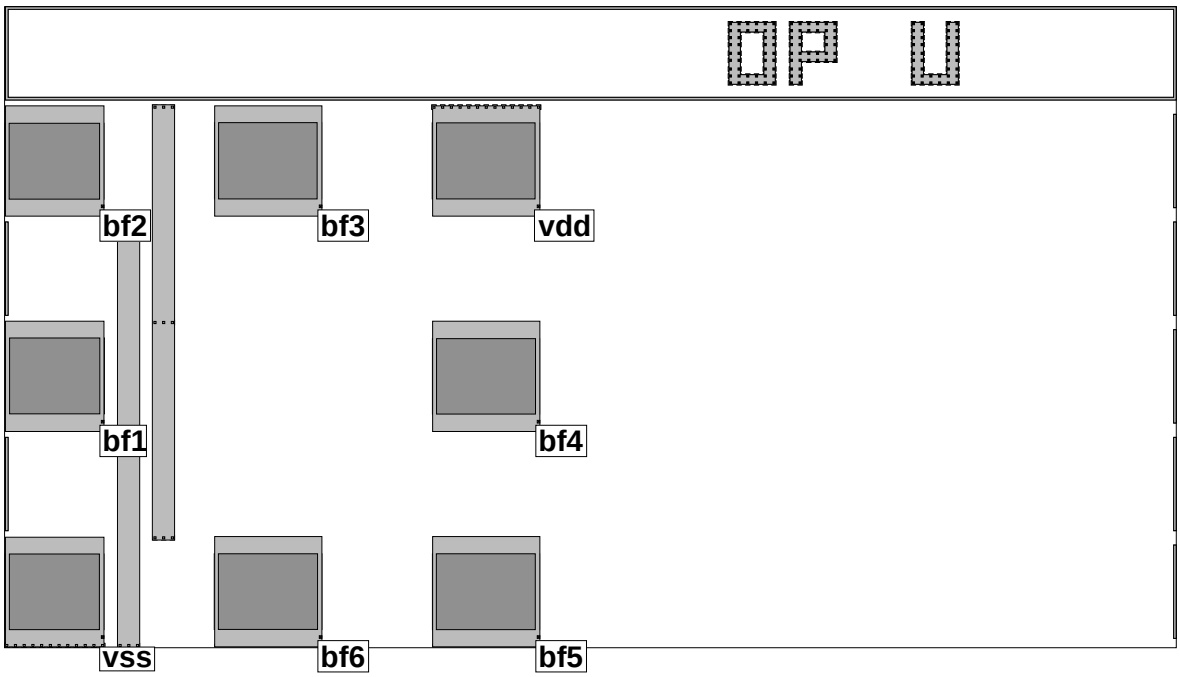
Maximum grootte schakeling: 350

Aansluitingen:

- Aantal: 8, waarvan 6 vrij te kiezen, 2 voeding (vss, vdd)
- Soort beveiliging: protectiedioden, geen buffers

Meetmogelijkheden: 8-pins probe-kaart

Layout:



C.2.19 bond_bar

Functie: Basiscel voor kwartindeling SoG-chip

Terminals: (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28, t29, t30, t31, t32)

Equivalent chip oppervlak: 25000

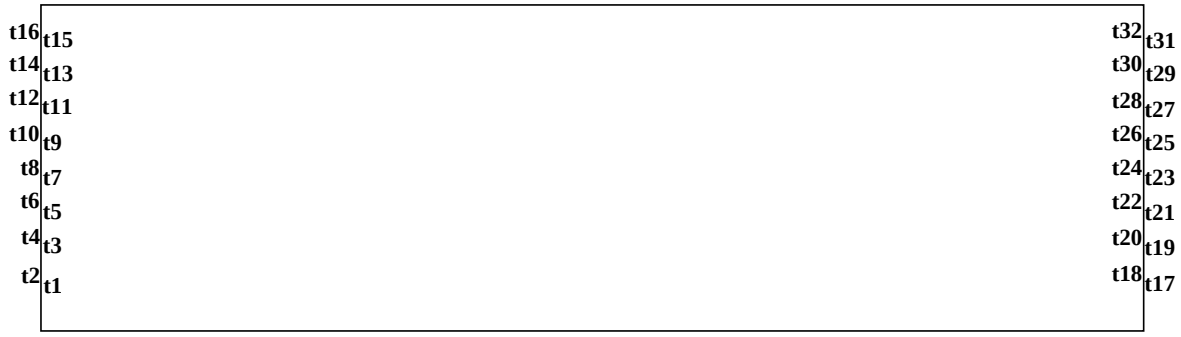
Maximum grootte schakeling: 20000

Aansluitingen:

- Aantal: 36, waarvan 32 vrij te kiezen, 2 voeding (chip), 2 voeding (buffers)
- Soort beveiliging: protectiedioden, buffers (in, out, direct)

Meetmoglijkheden: 36-pins probe-kaart, afmontage DIL-40

Layout:



D Introductie Sequentiële Machines

D.1 Inleiding

De Hotelschakeling, die in paragraaf 7.3 besproken wordt, is een voorbeeld van een *sequentiële schakeling*: een schakeling met geheugenwerking. Bij een sequentiële schakeling hangen de uitgangssignalen niet alleen af van de ingangssignalen: de schakeling kan, afhankelijk van de *toestand* waarin hij zich bevindt, verschillende uitgangssignalen produceren bij dezelfde ingangssignalen. Dit in tegenstelling tot combinatorische logica, waarbij gegeven ingangssignalen altijd dezelfde uitgangssignalen zullen opleveren. Sequentiële schakelingen onderscheiden zich van combinatorische logica doordat ze geheugenelementen bevatten. Deze geheugenelementen kunnen latches zijn (bij sequentiële schakelingen volgens het *level mode* model), of flipflops (*clock mode* model). In de geheugenelementen wordt de toestand van de schakeling opgeslagen. Sequentiële schakelingen kunnen beschreven worden d.m.v. een toestandsdiagram.

Bij het practicum zullen we sequentiële schakelingen nodig hebben om besturingseenheden te realiseren. Om het ontwerpen van deze besturingseenheden gestructureerd te laten verlopen en de verificatie van de ontworpen schakelingen te vereenvoudigen zullen we een aantal afspraken maken, waaraan de besturingen moeten voldoen:

- We maken gebruik van een één-fase kloksignaal. De toestand van de schakeling zal bewaard worden in flipflops, die één keer per klokperiode inlezen.
- Alleen schakelingen volgens het *Moore model* zijn toegestaan. Dit betekent, kort gezegd, dat de uitgangssignalen niet onmiddellijk af mogen hangen van de ingangssignalen, maar alleen van de toestand waarin de schakeling zich bevindt.
- De schakelingen moeten volledig *synchroon* zijn. Dat wil zeggen dat alle geheugenelementen op hetzelfde tijdstip inlezen. Het is dus niet toegestaan om logica op te nemen in de kloklijnen, omdat dit tot klokverschuiving (*clock-skew*) leidt. Ook het gebruik van ongelijke clock-buffers kan clock-skew veroorzaken.
- Alle ongebruikte toestanden moeten gecodeerd worden. Dit om te voorkomen dat de schakeling kan vastlopen in één of meerdere ongebruikte toestanden.

In de volgende paragrafen zullen we een praktische inleiding geven in het beschrijven van sequentiële schakelingen. Een uitgebreidere en meer formele behandeling is te vinden in [2].

D.2 Het Moore model

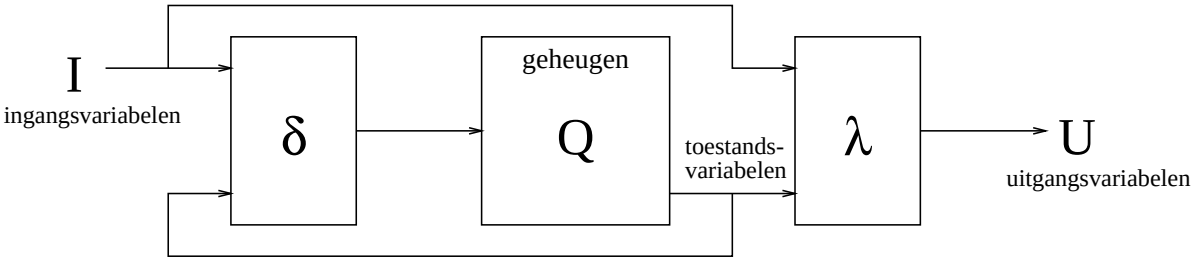
Voordat we kunnen gaan kijken wat het Moore model nu precies inhoudt zullen we eerst het algemene wiskundige model van een sequentiële schakeling bekijken: de *sequentiële machine*. Deze kan gedefinieerd worden als een geordend vijftal:

$$M = (I, U, Q, \delta, \lambda) \tag{1}$$

waarin

I de verzameling van *ingangssymbolen* is;
U de verzameling van *uitgangssymbolen* is;
Q de verzameling van *toestanden* is;
δ de *toestandsfunctie*, een afbeelding van $Q \times I$ in Q is;
λ de *uitgangsfunctie*, een afbeelding van $Q \times I$ op U is.

We zullen hierbij aannemen dat de verzamelingen I, U en Q eindig en niet-leeg zijn. Bovenstaande definitie is schematisch weergegeven in figuur 35.



Figuur 35: Blokschema sequentiële machine (Mealy model).

De blokken die aangeduid zijn met δ en λ stellen combinatorische logica voor; het blok dat aangeduid is met Q bevat de geheugenelementen. Het model van figuur 35 staat in de literatuur bekend als het *Mealy model*. In dit model is de λ-functie gedefinieerd als een functie van (het Cartesisch produkt van) de toestandsverzameling Q en de ingangsverzameling I op de uitgangsverzameling U:

$$\lambda : Q \times I \stackrel{op}{\rightarrow} U \tag{2}$$

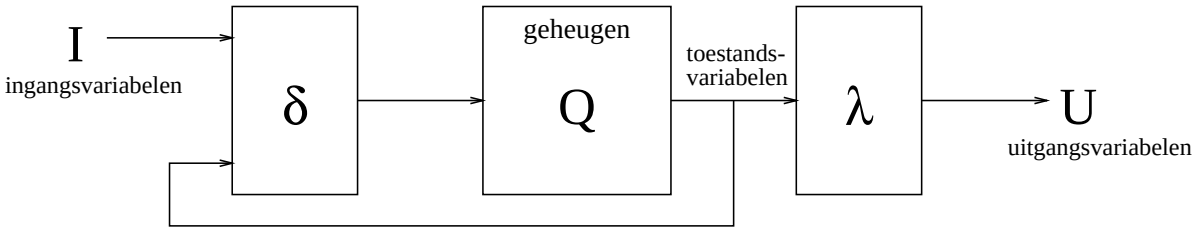
Het Mealy model staat dus een direct combinatorisch verband toe tussen de ingangswaarde en de uitgangswaarde. Dit betekent dat bij schakelingen volgens het Mealy model de uitgangen direct kunnen veranderen wanneer één van de ingangssignalen verandert.

Bij het zogenaamde *Moore model* is dit directe verband van ingang naar uitgang niet toegestaan. Het Moore model definieert de λ-functie als volgt:

$$\lambda : Q \stackrel{op}{\rightarrow} U \tag{3}$$

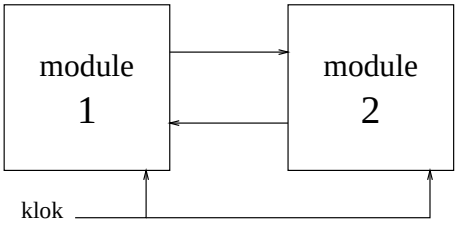
Dit is schematisch weergegeven in figuur 36.

Wiskundig gezien is het Moore model dus een beperking van het Mealy model. Bij het toepassen van beide modellen op praktische schakelingen blijkt dat schakelingen die volgens het Mealy model gespecificeerd zijn in het algemeen een kortere reactietijd hebben (in aantal klokperioden gerekend) dan schakelingen volgens het Moore model. Daarnaast vergt een specificatie volgens het Mealy model meestal minder toestanden. Een groot nadeel van het Mealy model is echter dat er zgn. *statische lusvorming* kan optreden. Dit kan gebeuren wanneer twee apart ontworpen schakelingen op elkaar worden aangesloten, zoals is weergegeven in figuur 37.



Figuur 36: Blokschema van het Moore model.

Figuur 37 toont twee modules die met elkaar communiceren onder besturing van een kloksignaal. Wanneer beide schakelingen zijn gespecificeerd volgens het Mealy model is het mogelijk dat een uitgangssignaal van module 1 rechtstreeks de uitgang van module 2 beïnvloedt. Het uitgangssignaal van module 2 gaat naar module 1 en kan zo op zijn beurt onmiddellijk het uitgangssignaal van module 1 veranderen. Dit kan weer zijn invloed hebben op de uitgang van module 2, enzovoort. Er is dan sprake van statische (= logische) lusvorming tussen module 1 en module 2.



Figuur 37: Interactie tussen twee modules.

Het is duidelijk dat er op deze manier een oscillatie kan ontstaan in de communicatie tussen beide modules. Het is in dat geval onzeker wat voor waarde de in- en uitgangssignalen van beide modules zullen hebben op het moment dat de eerstvolgende klokpuls komt. Hierdoor kan het gebeuren dat twee modules die elk op het eerste gezicht correct gespecificeerd lijken, toch niet goed functioneren wanneer ze aan elkaar gekoppeld worden.

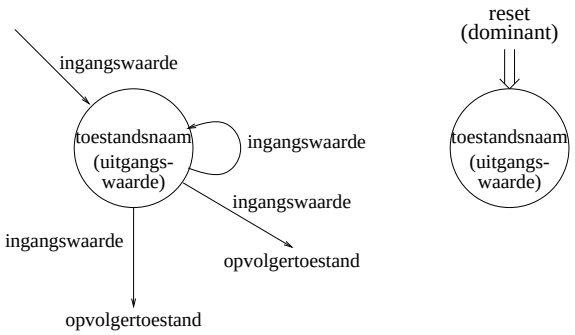
Wanneer beide modules daarentegen volgens het Moore model gespecificeerd zijn, zal het probleem van statische lusvorming zich niet voordoen. Wanneer module 1 zijn uitgang verandert, kan dit pas na de eerstvolgende klokpuls van invloed zijn op de uitgang van module 2. Hierop kan module 1 pas een klokpuls later reageren. De mogelijkheid van oscillaties binnen één klokperiode is hier dus niet aanwezig. Dit maakt de verificatie van schakelingen volgens het Moore model een stuk eenvoudiger dan schakelingen volgens het Mealy model. Daarom wordt voor het ontwerppracticum geëist dat alle sequentiële schakelingen van het Moore-type zijn.

D.3 Toestandsdiagrammen

Sequentiële schakelingen kunnen worden weergegeven door middel van toestandsdiagrammen. We zullen ons hier beperken tot sequentiële schakelingen volgens het Moore model.

In een toestandsdiagram geven we iedere toestand weer met een bolletje. In het bolletje wordt de naam van de toestand gezet. Ook kan de uitgangscombinatie die bij die toestand hoort in het bolletje gezet

worden. [1] Overgangen van de ene toestand naar de andere worden weergegeven met pijlen. Bij iedere pijl staat voor welke ingangscombinatie de overgang optreedt. De meeste besturingsschakelingen hebben één resettoestand en een dominant resetsignaal. [2] Wanneer het resetsignaal geactiveerd wordt, komt de besturing altijd in de resettoestand, ongeacht de andere ingangssignalen. De resettoestand kan in een toestandsdiagram aangegeven worden d.m.v. een dubbele pijl. De aparte pijlen vanuit iedere toestand naar de resettoestand kunnen dan weggelaten worden. Dit is weergegeven in figuur 38.



Figuur 38: Links: voorbeeld van een stukje toestandsdiagram. Rechts: aanduiding van een dominant resetsignaal.

Het is handig om in toestandsdiagrammen symbolische namen te geven aan toestanden: een toestandsdiagram met namen als 'rusttoestand' en 'wacht op input' is meestal duidelijker dan een toestandsdiagram met namen als 'Q0' en 'Q1'. Hetzelfde geldt voor de in- en uitgangssignalen: symbolische benamingen zijn meestal duidelijker dan rijen bits. In een later stadium wordt dan een toestandstabel opgesteld, waarin bitcombinaties worden toegekend aan de symbolische namen. Ook is het mogelijk om in eerste instantie een groep toestanden weer te geven als één enkele toestand, en deze groep pas in een later stadium verder uit te werken: zulke toestanden worden *meta-toestanden* genoemd. Een andere reden om een toestand op te splitsen in een aantal toestanden kan bijvoorbeeld zijn dat bij het uitwerken van een ontwerp blijkt dat een gevraagde actie niet in één klokperiode te realiseren is, en dus verdeeld moet worden over meerdere perioden.

D.4 Voorbeeld: Modeltrein-besturing

In deze paragraaf zullen we het ontwerp van een eenvoudige sequentiële schakeling uitwerken. De opdracht is als volgt:

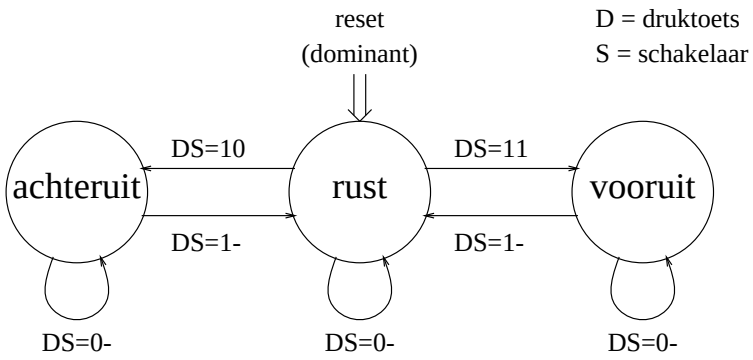
OPDRACHT
Ontwerp de besturing voor een modeltrein. De trein kan in twee richtingen rijden en wordt bediend d.m.v. een druktoets en een schakelaar. Met de druktoets kan de trein op gang of tot stilstand gebracht worden. De schakelaar bepaalt of de trein voor- of achteruit rijdt. De rijrichting kan alleen veranderd

[1] Bij het Moore model is de uitgang alleen afhankelijk van de toestand waarin de schakeling zich bevindt. Bij het Mealy model is de uitgang ook afhankelijk van de huidige ingangscombinatie, zodat de uitgangswaarden dan bij de pijlen voor de toestandsovergangen komen te staan.

[2] Onder een dominant resetsignaal zullen we hier verstaan: een signaal dat de schakeling op de eerstvolgende klokpuls in één gedefinieerde toestand brengt, ongeacht de waarde van de overige ingangssignalen of de toestand waarin de schakeling zich bevindt.

worden door de trein eerst tot stilstand te brengen.
De beide ingangssignalen zijn gesynchroniseerd met de klok. Het indrukken van de druktoets levert een puls op ter lengte van één klokperiode. Er is een resetsignaal dat bij het aanzetten van de modeltrein gedurende enkele klokperioden hoog is, en daarna verder laag (power-on reset). De besturing heeft twee uitgangssignalen, die aangeven of de trein voor- of achteruit rijdt. Wanneer beide signalen laag zijn, staat de trein stil.

UITWERKING
De besturing heeft drie toestanden: rust, vooruit en achteruit. Overgangen tussen deze toestanden treden op wanneer de druktoets wordt ingedrukt. De stand van de schakelaar bepaalt of de trein bij het indrukken van de druktoets in de rusttoestand, vooruit (schakelaar = 1) of achteruit (schakelaar = 0) gaat rijden. Het power-on resetsignaal kan gebruikt worden om te voorkomen dat de trein meteen begint te rijden, wanneer de modelspoorbaan wordt aangezet. [3] Hiertoe wordt de rusttoestand als resettoestand gekozen. Het resetsignaal is dominant over de andere ingangssignalen. Het toestandsdiagram is weergegeven in figuur 39. Merk op dat er in het toestandsdiagram gebruik is gemaakt van het gegeven dat de druktoets pulsen afgeeft met een tijdsduur van precies één klokperiode. Wanneer een puls langer aanhoudt, zal de schakeling gaan oscilleren tussen de rusttoestand en één van de andere twee toestanden, afhankelijk van de stand van de schakelaar. Dit valt op te lossen door het toestandsdiagram uit te breiden met drie extra toestanden. De overgangen in het toestandsdiagram lopen dan alle via zo'n extra toestand, waarin gewacht wordt op het loslaten van de druktoets, alvorens door te gaan naar één van de huidige toestanden 'rust', 'vooruit' en 'achteruit'.



Figuur 39: Toestandsdiagram van de modeltrein-besturing.

We kunnen nu dit toestandsdiagram gaan vertalen naar een schakeling. De toestand van de schakeling kan bewaard worden in flipflops. Omdat er drie toestanden zijn, kunnen we de toestand coderen in twee flipflops: Q_1 en Q_0. Wanneer we flipflops gebruiken met een synchrone reset-ingang, kunnen we het resetsignaal hier meteen op aansluiten. Daarmee wordt de rusttoestand gecodeerd met $Q_1Q_0 = 00$. We kunnen vervolgens bitcombinaties toekennen aan de overige toestanden. We kunnen de combinaties 01 en 10 gebruiken om de toestanden 'achteruit' en 'vooruit' te coderen. We hebben dan nog één combinatie niet gebruikt: $Q_1Q_0 = 11$. We moeten zorgen dat, mocht de schakeling in deze toestand terechtkomen, hij er niet in vast blijft zitten, en dat de uitgangssignalen in ieder geval 'netjes' zijn gedefinieerd[4].

[3] Het is niet bekend wat voor waarde de geheugenelementen aannemen als de schakeling wordt aangezet. Daardoor is ook de toestand waarin de schakeling bij het aanzetten terecht komt onbekend.

[4] We kunnen in principe de toestandsvariabele Q_1 gebruiken om het uitgangssignaal 'vooruit' te generen, en Q_0 voor 'achteruit'. Dit heeft als voordeel dat er geen extra logica nodig is om de uitgangssignalen te bepalen, behalve een eventuele

Q_1	Q_0	toestand	Vooruit	Achteruit
0	0	rust	0	0
0	1	achteruit	0	1
1	0	vooruit	1	0
1	1	ongebruikt	0	0

Tabel 5: Toestandscodering en uitgangssignalen.

De codering van de toestanden met de bijbehorende uitgangssignalen is weergegeven in tabel 5. Tabel 6 geeft de bijbehorende functietabel, waarin de signalen druktoets, schakelaar, vooruit en achteruit zijn afgekort tot resp. D, S, V, A. Met de fase n of $n + 1$ van een signaal wordt de waarde die dat signaal aanneemt na de n^e resp. $n + 1^e$ klokpuls aangeduid. In de tabel is het resetsignaal weggelaten (dit wordt rechtstreeks aangesloten op de flipflops). Deze tabel kan worden omgezet naar combinatorische logica, om de opvolgertoestanden te bepalen (δ in figuur 36) en de uitgangssignalen te genereren (λ in figuur 36). Deze omzetting kan handmatig gedaan worden, of m.b.v synthese software. Dan kan er een netwerk aangemaakt worden, waarin twee flipflops gekoppeld worden aan de logica. Daarbij worden de uitgangen voor Q^{n+1} verbonden met de D-ingangen van de flipflops. De netwerkbeschrijving is dan gereed, en de besturingseenheid kan getest worden.

D^n	S^n	Q_1^n	Q_0^n	Q_1^{n+1}	Q_0^{n+1}	V^n	A^n
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
1	1	0	0	1	0	0	0
0	-	0	1	0	1	0	1
1	-	0	1	0	0	0	1
0	-	1	0	1	0	1	0
1	-	1	0	0	0	1	0
-	-	1	1	0	0	0	0

Tabel 6: Functietabel van de modeltrein-besturing.

buffer om de flipflops niet te zwaar te belasten. Wanneer de schakeling dan in de ongebruikte toestand komt, zullen beide uitgangen echter hoog worden. Hier zou de modeltrein kapot van kunnen gaan. Daarom is er hier voor gekozen om beide uitgangen in deze toestand laag te maken, hetgeen ten koste zal gaan van wat extra logica.

E Het schrijven van VHDL code voor simulatie en synthese

E.1 Inleiding

Tijdens het ontwerpproces is het belangrijk om VHDL code te schrijven die niet alleen de gewenste simulatieresultaten vertoont, maar die ook geschikt om door de logic synthesiser te worden omgezet in een schakeling die realiseerbaar is, en die daarna tijdens simulatie nog steeds het gewenste gedrag vertoont. In dit gedeelte zullen enkele richtlijnen worden gegeven om dit zo succesvol mogelijk te laten verlopen. Belangrijk hierbij is dat men zich tijdens het ontwikkelen van de VHDL code realiseert hoe de structuur van de schakeling zal zijn waarvoor men het gedrag beschrijft. Dit in ogenschouw nemende zullen we de volgende typen van VHDL code onderscheiden en apart behandelen.

- Gedragsbeschrijving combinatorische schakeling
- Gedragsbeschrijving sequentiële schakeling
- Structuurbeschrijving van een schakeling

Daarnaast zal voor sequentiële schakelingen nog worden verduidelijkt hoe alternatieve vormen van VHDL codering invloed hebben op simulatie en synthese.

E.2 Gedragsbeschrijving combinatorische schakelingen

Een gedragsbeschrijving van een combinatorische schakeling bestaat uit een architecture body met daarin concurrent statements zoals process statements en dataflow statements. Bij combinatorische schakelingen is het belangrijk om te zorgen dat elk uitgangsignaal een waarde krijgt voor elke combinatie van ingangsignalen. Hieronder volgt een voorbeeld van een combinatorische schakeling met een process statement.

```
library IEEE;
use IEEE.std_logic_1164.ALL;

architecture behaviour of combinatorial is
begin
   -- a, b and c are input signals, y is output signal
   lbl1: process (a, b, c)
   -- All input signals must be in the sensitivity list
   begin
      -- Compute output value(s) based on input values
      -- Important: for every input combination, each
      -- output should receive a value !
      if ((a = '1' or b = '1') and c = '0') then
         y <= '1';
      else
         y <= '0';
```

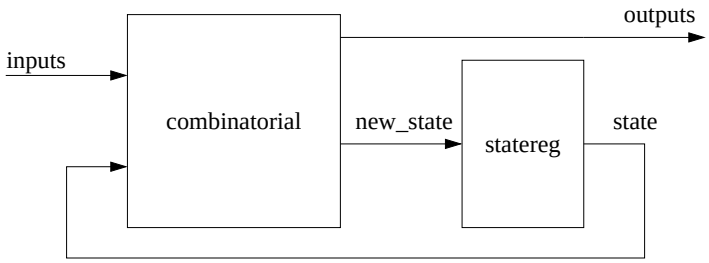
```
      end if;
   end process;
end behaviour;
```

Hier is dezelfde combinatorische schakeling op meer compacte wijze beschreven m.b.v. een dataflow statement:

```
architecture behaviour of combinatorial is
begin
   y <= (a or b) and (not c);
end behaviour;
```

E.3 Gedragsbeschrijving sequentiële schakeling

We gaan er van uit dat elke sequentiële schakeling die we ontwerpen beschreven wordt door een Moore machine zoals in figuur 40. Zie ook figuur 36, waarbij δ en γ zijn samen genomen in een "combinatorial" blok, en Q gelijk is aan het "statereg" blok. Bij het beschrijven van een sequentiële schake-



Figuur 40: Blokschema voor het Moore model.

ling maken we daarom onderscheid tussen een register gedeelte dat de toestand onthoudt ("statereg") en een combinatorisch gedeelte dat de nieuwe toestand en de waarden van de uitgangsignalen berekent aan de hand van de ingangsignalen en de huidige state ("combinatorial"). Hoe de combinatorische schakeling "combinatorial" beschreven kan worden hebben we gezien in de voorafgaande sectie. Bij de beschrijving van het gedeelte dat de toestand onthoudt moeten we ons realiseren dat we in de Sea-of-Gates cellenbibliotheek de beschikking hebben over positive-edge triggered flipflops. Deze flipflops zijn uitgevoerd zonder reset, met een synchrone reset of met een asynchrone reset. Ook op het Altera bord bevinden zich flipflops van deze types. Wanneer "state" de huidige toestand weergeeft, "new_state" de volgende toestand, "RESET_STATE" de reset toestand, "clk" het klok signaal en "res" het reset signaal, en wanneer we uitgaan van een synchrone reset, dan kunnen we het gedeelte voor het opslaan van de toestand beschrijven met

```
   statereg: process (clk)
   begin
      if (clk'event and clk = '1') then
         if res = '1' then
            state <= RESET_STATE;
         else
```

```
            state <= new_state;
         end if;
      end if;
   end process;
```

Een compleet voorbeeld voor een beschrijving voor een sequentiële schakeling wordt dan:

```
library IEEE;
use IEEE.std_logic_1164.ALL;

architecture behaviour of fsm is
   type state_type is (STATE1, STATE2);
   signal state, new_state: state_type;
begin
   statereg: process (clk)
   begin
      -- Because we use positive-edge triggered memory elements
      -- (flipflops) with synchronous reset, we assign a new
      -- state when clock goes to '1'
      if (clk'event and clk = '1') then
         if res = '1' then
            state <= STATE1;     -- assign reset state
         else
            state <= new_state;
         end if;
      end if;
   end process;
   combinatorial: process (state, input1)
   begin
      -- Assign output values based on state only (Moore machine)
      -- and compute new_state based on current state and input
      -- signals. Important: for all possible states, every output
      -- should receive a value and new_state should be defined.
      case state is
         when STATE1 =>
            output1 <= '0';
            if (input1 = '0') then
               new_state <= STATE2;
            else
               new_state <= STATE1;
            end if;
         when STATE2 =>
            output1 <= '1';
            new_state <= STATE1;
      end case;
   end process;
end behaviour;
```

Wanneer we zouden kiezen voor een asynchrone reset dan zou het eerste process er als volgt uitzien:

```
   statereg: process (clk, res)
   begin
      -- State register using positive-edge triggered memory elements
      -- and an asynchronous reset (note that res is in sensitity list).
      if res = '1' then
         -- assign reset state
         state <= STATE1;
      elsif (clk'event and clk = '1') then
         state <= new_state;
      end if;
   end process;
```

Een ander voorbeeld van een sequentiële schakeling die op bovenstaande wijze beschreven is, is de hotel schakeling die als voorbeeld wordt gegeven bij de inwerkopdrachten. Nog een ander voorbeeld is een 4 bits teller zoals die hieronder is gegeven. Merk op dat in dit geval de toestand wordt gevormd door de 4 bits van de std_logic_vector count en dat de uitgangen gevormd worden door de 4 bits van de vector a die hier rechtstreeks van worden afgeleid.

```
library IEEE;
use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.ALL;

architecture behaviour of counter is
   signal count, new_count : std_logic_vector(3 downto 0);
begin
   statereg: process (clk)
   begin
      if (clk'event and clk = '1') then
         if reset = '1' then
            count <= (others => '0');
         else
            count <= new_count;
         end if;
      end if;
   end process;
   combinatorial: process (count, enable)
   begin
      a <= std_logic_vector(count);
      if (enable = '1') then
         new_count <= count + 1;
      else
         new_count <= count;
      end if;
```

```
   end process;
end behaviour;
```

E.4 Structuurbeschrijving van een schakeling

Een structuurbeschrijving kan gezien worden als een fysiek netwerk/circuit van componenten en de verbindingen daartussen. Afhankelijk van welke componenten er in zijn opgenomen zal de structuurbeschrijving een combinatorische of een sequentiële schakeling beschrijven. De componenten zijn representaties voor entities en architecturen van andere delen uit het ontwerp. Eventueel kunnen kunnen de componenten ook cellen uit de Sea-of-Gates cellenbibliotheek zijn, wanneer de beschrijving alleen voor de Sea-of-Gates chip bedoeld is en niet op het Altera bord wordt afgebeeld. Het is niet nodig om een structuurbeschrijving te synthetiseren. Wanneer ook alle componenten op structuurniveau bekend zijn, dan kan direkt vanuit de structuurbeschrijving een layout aangemaakt worden.

Hieronder volgt een voorbeeld van een structuurbeschrijving. Naast het tekstueel invoeren van de VHDL code kan een VHDL structuurbeschrijving ook op eenvoudige wijze worden aangemaakt met een schematic editor zoals Schentry.

```
library IEEE;
use IEEE.std_logic_1164.ALL;

architecture circuit of network is
   -- First the declaration of the components used
   component comp1
      port (x: in std_logic; y: out std_logic);
   end component;
   component comp2
      port (p, q: in std_logic; r: out std_logic);
   end component;
   -- second the declaration of the nets (internal nodes)
   signal net1, net2, net3 : std_logic;
begin
   -- connections between nets and pins
   output1 <= net2;
   -- instances of components and their connections
   comp1_1: comp1 port map (x=>net1, y=>net3);
   comp1_2: comp1 port map (x=>net2, y=>net1);
   comp2_1: comp2 port map (p=>net3, q=>input1, r=>net2);
end circuit;
```

E.5 Verschillende vormen van VHDL beschrijvingen voor sequentiële schakelingen bekeken

Naast de vorm die in de OP handleiding aangeraden wordt voor het beschrijven van sequentiële schakeling zijn er meer vormen mogelijk. Deze andere vormen hebben echter andere eigenschap-

pen tijdens simulatie en/of synthese waarbij rekening gehouden moet worden wanneer ze gebruikt worden. Hieronder volgen enkele van deze alternatieve vormen voor het beschrijven van sequentiële schakeling, waarbij telkens de afwijkende eigenschappen beschreven worden. Uitgegaan wordt van het voorbeeld van de hotel schakelaar.

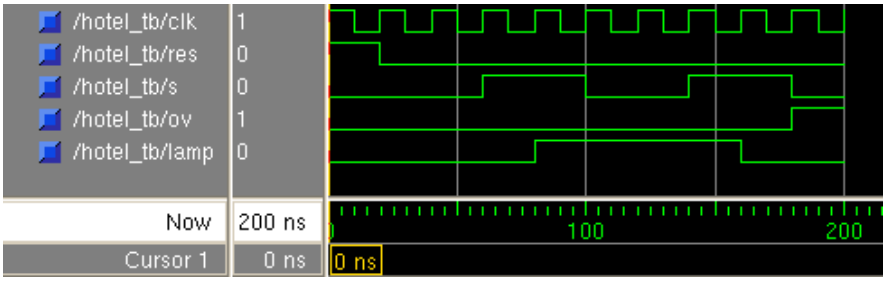
E.5.1 Aangeraden vorm

```
architecture behaviour of hotel is
   type lamp_state is (OFF0, OFF1, ON0, ON1);
   signal state, new_state: lamp_state;
begin
   lbl1: process (clk)
   begin
      if (clk'event and clk = '1') then
         if res = '1' then
            state <= OFF0;
         else
            state <= new_state;
         end if;
      end if;
   end process;
   lbl2: process(state, s, ov)
   begin
      case state is
         when OFF0 =>
            lamp <= '0';
            if (s = '1') and (ov = '0') then
                new_state <= ON1;
            else
                new_state <= OFF0;
            end if;
         when ON1 =>
            lamp <= '1';
            if (s = '0') and (ov = '0') then
                new_state <= ON0;
            else
                new_state <= ON1;
            end if;
         when ON0 =>
            lamp <= '1';
            if (s = '1') and (ov = '0') then
                new_state <= OFF1;
            else
                new_state <= ON0;
            end if;
         when OFF1 =>
            lamp <= '0';
            if (s = '0') and (ov = '0') then
                new_state <= OFF0;
            else
                new_state <= OFF1;
```

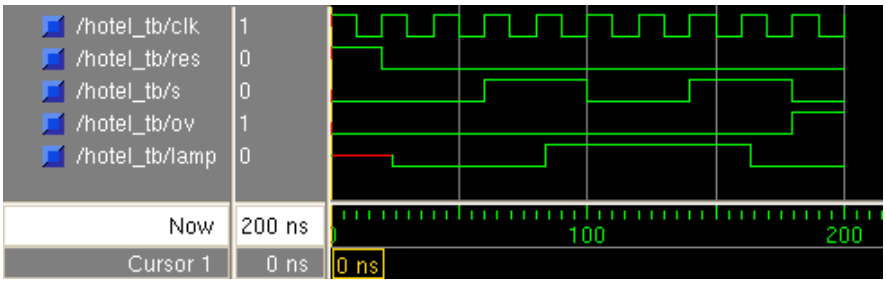
```
            end if;
        end case;
    end process;
end behaviour;
```

Bij deze vorm wordt er een process blok gebruikt voor het beschrijven van het (synchrone) toestandsregister en een process blok voor het beschrijven van de (asynchrone) schakeling die nodig is om - gebaseerd op de huidige toestand en de ingangssignalen - de nieuwe toestand en de uitgangssignalen te berekenen. Deze vorm wordt aangeraden omdat het een hoog niveau gedragsbeschrijving is die door synthesizers eenvoudig (zonder af te wijken van het oorspronkelijke gedrag) is om te zetten naar een schakeling bestaande uit flipflops en logische poorten. Deze vorm leidt, na synthese voor de Sea-of-Gates bibliotheek, tot een schakeling met 2 flipflops.

De simulatie van bovenstaande gedragsbeschrijving (Figuur 41) en de simulatie resultaten van het gesynthetiseerde circuit (Figuur 42) geven eenzelfde resultaat, afgezien van een klein verschil in vertragingstijden (bij de gesynthesieerde schakeling zijn de poort vetragingstijden meegenomen) en van de waarde van uitgangssignaal in de initialisatie-fase.



Figuur 41: Simulatie behavior beschrijving



Figuur 42: Simulatie gesynthetiseerde schakeling

E.5.2 Berekening van nieuwe toestand binnen het geklokte process blok

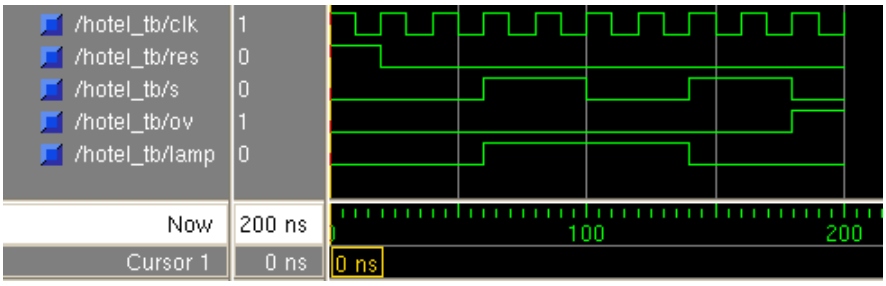
Bij de volgende vorm van VHDL coderen is in het eerste (geklokte) process blok, naast de toekenning van de nieuwe toestand, ook de berekening van de nieuwe toestand opgenomen. In het tweede process is alleen gespecifeerd hoe de uitgang van de schakeling afhangt van de huidige toestand.

```
architecture behaviour of hotel is
```

```
   type lamp_state is (OFF0, OFF1, ON0, ON1);
   signal state: lamp_state;
begin
   lbl1: process (clk)
   begin
      if (clk'event and clk = '1') then
         if (res = '1') then
            state <= OFF0;
         else
            case state is
               when OFF0 =>
                  if (s = '1') and (ov = '0') then
                     state <= ON1;
                  else
                     state <= OFF0;
                  end if;
               when ON1 =>
                  if (s = '0') and (ov = '0') then
                     state <= ON0;
                  else
                     state <= ON1;
                  end if;
               when ON0 =>
                  if (s = '1') and (ov = '0') then
                     state <= OFF1;
                  else
                     state <= ON0;
                  end if;
               when OFF1 =>
                  if (s = '0') and (ov = '0') then
                     state <= OFF0;
                  else
                     state <= OFF1;
                  end if;
            end case;
         end if;
      end if;
   end process;
   lbl2: process (state)
   begin
      if (state = OFF0) or (state = OFF1) then
         lamp <= '0';
      else
         lamp <= '1';
      end if;
   end process;
end behaviour;
```

De simulatie resultaten van de behavior beschrijving zijn in dit geval afwijkend van die voor de aangeraden behavior beschrijving, zie Figuur 43. De verklaring hiervoor is als volgt. Omdat in de testbench het ingangssignaal s verandert op het moment dat ook de klok omhoog gaat, zal deze "nieuwe" waarde in het eerste process blok van de behavior beschrijving worden meegnomen in de

berekening van de nieuwe toestand op het moment dat de klok omhoog gaat.



Figuur 43: Simulatie behavior beschrijving

In de fysische werkelijkheid zal bij het tegelijkertijd veranderen van klok en ingangssignaal (door de setup tijd van de flipflop) de “oude” waarde van het inganssignaal worden gebruikt voor het berekenen van de nieuwe toestand, en wordt de de “nieuwe” waarde gebruikt tijdens de volgende opgaande klokflank. Het simulatie resultaat van de schakeling die onstaat na synthese voor de vorm die hier beschreven wordt, is dan ook wel gelijk aan het simulatie resultaat voor de aangeraden vorm. Verder zal ook hier tijdens synthese een schakeling worden gegenereerd met 2 flipflops.

Wanneer er voor gezorgd wordt dat de ingangssignalen nooit tegelijkertijd met de opgaande klokflank veranderen zal bovengenoemd simulatie verschil uiteraard niet optreden en zal de hier beschreven gedragsbeschrijving dezelfde resultaten geven (voor simulatie en synthese) als de aangeraden vorm.

E.5.3 Berekening van nieuwe toestand en uitgangssignalen binnen het geklokte process blok

```
architecture behaviour of hotel is
   type lamp_state is (OFF0, OFF1, ON0, ON1);
   signal state: lamp_state;
begin
   lbl1: process (clk)
   begin
      if (clk'event and clk = '1') then
         if (res = '1') then
            state <= OFF0;
            lamp <= '0';
         else
            case state is
               when OFF0 =>
                  if (s = '1') and (ov = '0') then
                     state <= ON1; lamp <= '1';
                  else
                     state <= OFF0; lamp <= '0';
                  end if;
               when ON1 =>
                  if (s = '0') and (ov = '0') then
                     state <= ON0;
                  else
                     state <= ON1;
```

```
                end if;
                lamp <= '1';
             when ON0 =>
                if (s = '1') and (ov = '0') then
                   state <= OFF1; lamp <= '0';
                else
                   state <= ON0; lamp <= '1';
                end if;
             when OFF1 =>
                if (s = '0') and (ov = '0') then
                   state <= OFF0;
                else
                   state <= OFF1;
                end if;
                lamp <= '0';
          end case;
       end if;
    end if;
  end process;
end behaviour;
```

Bij deze vorm vindt zowel de berekening van de nieuwe toestand als de berekening van de uitgangssignalen plaats in het geklokte process blok. Hoewel dit leidt tot een compacte beschrijving met slechts 1 process blok, moet wel rekening worden gehouden met: (1) hetzelfde afwijkende simulatie gedrag van de behavior beschrijving als bij de hiervoor beschreven vorm, en (2) het feit dat de synthesizer nu een schakeling met 3 flipflops zal genereren i.p.v. 2. Dit laatste komt doordat de synthesizer behalve het signal "state" ook het signal "lamp" als een onderdeel van de toestand opvat, en dus naast 2 flipflops voor "state" ook 1 flipflop voor "lamp" gebruikt.

E.5.4 Het lezen van signalen die ook geschreven worden binnen een process blok

```
architecture behaviour of hotel is
  type lamp_state is (OFF0, OFF1, ON0, ON1);
  signal state: lamp_state;
begin
  lbl1: process (clk)
  begin
    if (clk'event and clk = '1') then
      if (res = '1') then
        state <= OFF0;
      else
        case state is
          when OFF0 =>
            if (s = '1') and (ov = '0') then
              state <= ON1;
            else
              state <= OFF0;
            end if;
          when ON1 =>
            if (s = '0') and (ov = '0') then
```

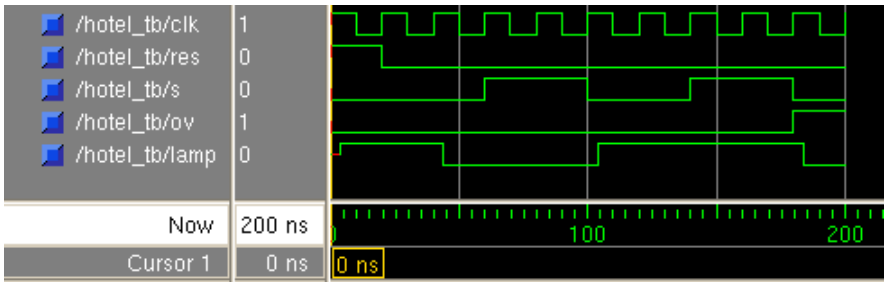
```
                  state <= ON0;
               else
                  state <= ON1;
               end if;
            when ON0 =>
               if (s = '1') and (ov = '0') then
                  state <= OFF1;
               else
                  state <= ON0;
               end if;
            when OFF1 =>
               if (s = '0') and (ov = '0') then
                  state <= OFF0;
               else
                  state <= OFF1;
               end if;
         end case;
      end if;
      if (state = ON0) or (state = ON1) then
          lamp <= '1';
      else
          lamp <= '0';
      end if;
   end if;
end process;
end behaviour;
```

Hierbij is het simulatie resultaat van de gedragsbeschrijving gelijk aan dat van de aangeraden vorm. De verklaring hiervoor is als volgt. Net zoals in de voorafgaande vorm zal de toestand "state" onmiddelijk ge-update wordt met de "nieuwe" waarde van het ingangssignaal en zorgt dit voor 1 klokpuls verschuiving vooruit in de tijd van het het toestandssignaal. Echter, in het if-statement dat de waarde voor lamp berekent, wordt de oude waarde van state gebruikt (dit is een belangrijke algemene eigenschap die geldt voor signals in VHDL beschrijvingen: de nieuwe waarde die voor een signal berekend wordt, wordt pas toegekend tijdens een volgende uitvoering van het process blok !). Daardoor onstaat er 1 klokpuls vertraging voor de waarde van lamp (die ook in dit geval zal worden opgeslagen in een aparte 3e flipflop). Het geheel vertoont daardoor eenzelfde gedrag als de behavior beschrijving van de aangeraden schakeling.

Bij simulatie van het gesynthetiseerde circuit wordt "state" weer - door de setup tijd van de flipflop - ge-update 1 klokpuls nadat het ingangssignaal verandert. De synthesizer zal verder - overeenkomstig de behavior beschrijving - een schakeling genereren waarbij de waarde van de lamp 1 klokpuls achter loopt bij de toestand waar hij eigenlijk bij hoort. De schakeling als geheel vertoont dan een gedrag waarbij de lamp 1 klokpuls later reageert dan bij de aangeraden schakeling, zie Figuur 44

E.5.5 Het gebruik van variabelen i.p.v. signalen binnen een process blok

```
architecture behaviour of hotel is
   type lamp_state is (OFF0, OFF1, ON0, ON1);
begin
```



Figuur 44: Simulatie gesynthetiseerde schakeling

```
lbl1: process (clk)
variable state: lamp_state;
begin
   if (clk'event and clk = '1') then
      if (res = '1') then
         state := OFF0;
      else
         case state is
            when OFF0 =>
               if (s = '1') and (ov = '0') then
                  state := ON1;
               else
                  state := OFF0;
               end if;
            when ON1 =>
               if (s = '0') and (ov = '0') then
                  state := ON0;
               else
                  state := ON1;
               end if;
            when ON0 =>
               if (s = '1') and (ov = '0') then
                  state := OFF1;
               else
                  state := ON0;
               end if;
            when OFF1 =>
               if (s = '0') and (ov = '0') then
                  state := OFF0;
               else
                  state := OFF1;
               end if;
         end case;
      end if;
      if (state = ON0) or (state = ON1) then
          lamp <= '1';
      else
          lamp <= '0';
      end if;
   end if;
```

```
   end process;
end behaviour;
```

Deze beschrijving is gelijk aan de voorafgaande beschrijving behalve dat “state” is gedeclareerd als een variable i.p.v. een signal.

In de voorafgaande vorm zagen we dat de nieuwe waarde van een signal zoals “state” pas beschikbaar is tijdens een volgende uitvoering van een process (op de opgaande klokflank) en zorgt dit voor 1 klokpuls vertraging op het uitgangssignaal. Wanneer we “state” declareren als een variable i.p.v. als een signal, dan treedt dit verschijnsel niet op. Bij een variable wordt een nieuw berekekende waarde in een process onmiddelijk gebruikt in de daaropvolgende statements. Gevolg is dat het simulatie resultaat van de gedragsbeschrijving dan 1 klokpuls voor loopt t.o.v. het simulatie resultaat van de de gedragsbeschrijving van de aangeraden vorm en dat het simulatie resultaat van het gesynthetiseerde circuit gelijk is aan dat van de aangeraden beschrijving.

Ook hier zullen weer 3 flipflops gegenereerd worden tijdens synthese van de schakeling.

F Analoge Deelschakelingen

F.1 Inleiding

Een kenmerk van een digitaal signaal is het feit dat slechts twee signaalniveaus van betekenis zijn voor de informatie-inhoud van dat signaal. Schakelingen die digitale signalen verwerken worden digitale schakelingen genoemd. Soms is het zinnig om voor de codering van informatie in een signaal meer dan twee niveaus te gebruiken. Het maximale aantal signaalniveaus dat nuttig gebruikt kan worden, hangt af van het maximaal toegelaten signaalniveau en het kleinst onderscheidbare verschil tussen twee signaalniveaus. Een veel gebruikte maat hiervoor is het dynamische bereik van een signaal. Schakelingen die dit soort signalen verwerken heten analoge schakelingen.
Elektronische schakelingen zullen bij het verwerken van signalen altijd het signaal in zekere mate aantasten. Het is de taak van de ontwerper om de schakelingen zo te ontwerpen dat de schade tot een minimum beperkt blijft. Bij digitale schakelingen zijn er slechts twee signaalniveaus van belang. Alle tussenliggende signaalniveaus worden slechts "gepasseerd" tijdens de overgang van het ene bekende naar het andere bekende niveau. Hoe de digitale schakeling van het ene niveau omschakelt naar het andere is niet aan erg gecompliceerde eisen gebonden. In het algemeen is het feit dat een digitale schakeling binnen een bepaalde tijd schakelt een voldoende garantie voor een goede werking. Dit heeft tot gevolg dat de modellering van de schakeling op dit gebied erg simpel kan zijn. Een model hoeft tijdens het omschakelen niet nauwkeurig te beschrijven op welk tijdstip een bepaald tussenliggend signaalniveau gepasseerd wordt, daar deze signaalniveaus toch geen informatie bevatten.
Bij analoge signalen en circuits is dit anders. Daar worden alle mogelijke signaalniveaus gebruikt om de informatie in het signaal te coderen. De modellering van analoge circuits zal daarom gecompliceerder zijn dan die van digitale circuits. Dit heeft tot gevolg dat op plaatsen in het ontwerptrajekt waar de modellering ter sprake komt het ontwerptrajekt voor analoge schakelingen zal afwijken van dat voor digitale schakelingen. Het betekent echter ook dat voor een zeer groot gedeelte van het ontwerptrajekt er geen onderscheid gemaakt hoeft te worden tussen analoge en digitale schakelingen. Het is eigenlijk ook niet verstandig om schakelingen het label analoog of digitaal te geven. Het gaat om het karakter van het signaal dat verwerkt wordt of de schakeling die het verwerkt analoog of digitaal genoemd moet worden. Ontwerphulpmiddelen zoals layout editors, design-rule checkers, routers, die geen weet hebben van het karakter van de signalen die verwerkt worden, kunnen met evenveel gemak op analoge als op digitale schakelingen worden toegepast. Alleen hulpmiddelen die zwaar leunen op de modellering van de schakelingen zullen voor analoge schakelingen duidelijk anders zijn dan voor digitale. De circuit simulator is daarvan wel het meest sprekende voorbeeld. Digitale schakelingen zijn meestal groot van omvang. Om zeer grote schakelingen nog efficiënt te kunnen simuleren is het belangrijk dat de modellering van de componenten waaruit de schakeling is opgebouwd zo simpel mogelijk is. Hierbij kan met veel succes gebruik gemaakt worden van het feit dat de te verwerken signalen digitaal zijn en dat er dus maar twee signaalniveaus onderscheiden hoeven te worden. Simulatoren voor digitale schakelingen zullen dus gekenmerkt zijn door de toepassing van simpele modellen op grote hoeveelheden componenten. Door de simpele modellen die gebruikt worden, zijn ze echter meestal niet geschikt voor het nauwkeurig simuleren van analoge schakelingen. Voor analoge schakelingen zal daarom gebruikt gemaakt worden van een simulator die werkt met gecompliceerde, nauwkeurige modellen. Dit beperkt dan weer de grootte van de schakeling die nog efficiënt gesimuleerd kan worden. Gelukkig zijn analoge schakelingen meestal veel kleiner dan digitale, zodat de simulator toch efficiënt gebruikt kan worden.

F.2 Analoge signaalbewerkingen in de groepsopdracht

Inleiding Tijdens het ontwerp van de groepsopdracht zullen er in een aantal gevallen analoge signalen bewerkt of gegenereerd moeten worden. Deze signalen worden vaak opgedrongen door de "buitenwereld", omdat in de natuur bijna alle signalen meer dan twee niveaus kunnen aannemen. In deze paragraaf zullen we een aantal analoge signaalbewerkingen op een rijtje zetten die u kunt tegenkomen bij het maken van uw ontwerp.

Uitlezen van een toetsenbord Een toetsenbord bestaat uit een aantal schakelaars die open of gesloten kunnen zijn. De impedantie van een schakelaar die gesloten is, is lager dan van een schakelaar die open is. In het geval van het folie-toetsenbord dat voor het practicum beschikbaar is, is de impedantie in gesloten toestand ongeveer 100 Ω en in open toestand enkele MΩ's. De impedantie van een schakelaar zal moeten worden omgezet in een digitaal signaal. Hier vindt dus een eenvoudige één-bit AD-conversie plaats.

Generatie van sinusvormige signalen Omdat een digitaal signaal slechts twee niveaus kan aannemen, zal zo'n signaal er altijd blokvormig uitzien. De "beste" benadering van een sinus met een digitaal signaal is een blokgolf met een duty-cycle van 50%. Als we een dergelijke blokgolf met een amplitude van van $\frac{\pi}{4}E_0$ opdelen in zijn verschillende frequentie-componenten, krijgen we:

$$\frac{\pi}{4}E_0\mathbf{B}(\omega t) = E_0 sin(\omega t) + \frac{E_0}{3}sin(3\omega t) + \frac{E_0}{5}sin(5\omega t) + \frac{E_0}{7}sin(7\omega t) + \ldots. \tag{4}$$

De harmonische distorsie van dit signaal is de som van het vermogen van de hogere harmonischen gedeeld door de grondharmonische. Dus:

$$\frac{\left(\frac{E_0}{3}\right)^2 + \left(\frac{E_0}{5}\right)^2 + \left(\frac{E_0}{7}\right)^2 + \ldots}{{E_0}^2} \tag{5}$$

en dit is gelijk aan:

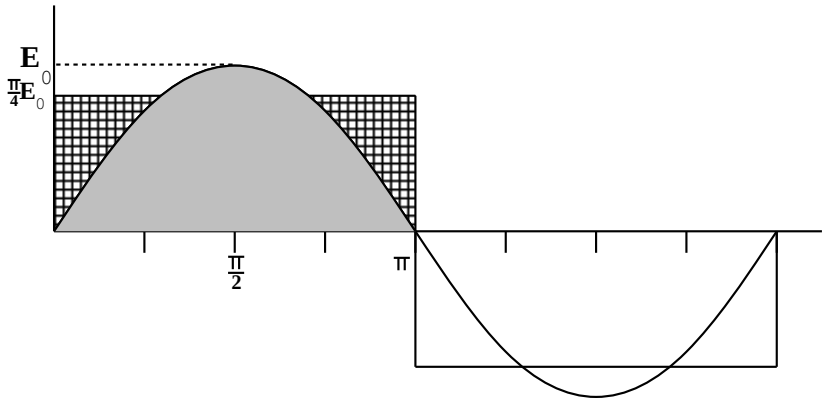
$$\left(\frac{1}{3}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{7}\right)^2 + \ldots = \sum_{n=1}^{\infty} \frac{1}{(2n+1)^2} \tag{6}$$

Dit kunnen we ook schrijven als:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} - \sum_{n=1}^{\infty} \frac{1}{2n^2} - 1 = \frac{\pi^2}{6} - \frac{\pi^2}{24} - 1 = \frac{\pi^2}{8} - 1 \approx 0.23 \tag{7}$$

Op een iets meer intuïtieve manier kunnen we dit ook zien aan de hand van figuur 45:

De harmonische distorsie is, zoals al eerder gezegd, de som van het vermogen van de hogere harmonischen gedeeld door de grondharmonische. Het vermogen van de grondharmonische is in de



Figuur 45: Figuur ter verduidelijking van bewijs harmonische distorsie blokgolf

figuur de oppervlakte onder de sinus. De som van de vermogens van de hogere harmonischen is de oppervlakte onder de blokgolf minus de oppervlakte onder de sinus. De harmonische distorsie is dus:

$$\frac{opp.\,blok\ -\ opp.\,sinus}{opp.\,sinus} = \frac{\pi\,.\,\frac{\pi}{4}E_0\ -\ \int_0^\pi E_0\,sin\,x\,dx}{\int_0^\pi E_0\,sin\,x\,dx} = \frac{\frac{\pi^2}{4}\ -\ 2}{2} = \frac{\pi^2}{8}\ -\ 1 \qquad (8)$$

Als we dus een signaal willen genereren met een lagere harmonische distorsie dan 0.23 of -6.4 dB, kan dat niet met een digitale schakeling.
Er zijn verschillende mogelijkheden om een sinusvormig signaal te genereren, b.v.:

- Met behulp van oscillatoren die elektronisch afstembaar zijn (Voltage Controlled Oscillator = VCO). Het digitale deel zal dan via een DA-converter (digitaal-analoog omvormer) de oscillator op de juiste frequentie afstemmen.

- Met behulp van een teller en een DA-converter. De teller zal een DA-converter zodanig aansturen dat aan de uitgang van de DA-converter een sinusvormig signaal verschijnt van de juiste frequentie.

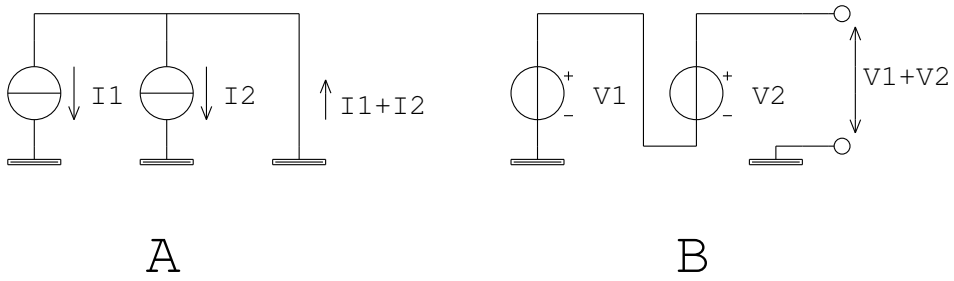
- Door het digitale circuit een blokvormig (digitaal) signaal van de juiste frequentie te laten leveren wat daarna gefilterd moet worden om hogere harmonischen in het signaal te verwijderen.

- Met behulp van puls-breedte modulatie. Een hoogfrequent draaggolf wordt puls-breedte gemoduleerd met een gekwantificeerde sinus. De draaggolf wordt vervolgens uitgefilterd.

Versterken Versterken is de meest gebruikte analoge signaalbewerking. Aan de ingang van een ontwerp is meestal versterking nodig omdat het inkomende signaal vaak te zwak is om direct gedigitaliseerd te worden. Aan de uitgang omdat het digitale signaal vaak te weinig vermogen heeft om direct een actuator [5] aan te sturen.

[5] Voorbeelden van actuatoren zijn: een luidspreker, robot-arm, beeldbuis, enz.

Optellen Het optellen van twee signalen kan zowel digitaal als analoog gebeuren. In dit hoofdstuk zullen we ons beperken tot de analoge methode. In dit laatste geval worden de twee op te tellen signalen eerst afzonderlijk gegenereerd en daarna opgeteld. De twee gegenereerde signalen zullen beschikbaar zijn als spanning of als stroom. Stromen kunnen vrij simpel worden opgeteld door ze simpelweg in hetzelfde knooppunt te laten vloeien (parallel schakeling). Door de uitgangen van twee stroombronnen die ieder een sinusvormige stroom produceren met elkaar te verbinden is de optelling gerealiseerd.
Om spanningen te kunnen optellen moeten de spanningsbronnen in serie geschakeld worden. Dit betekent dat tenminste één van de twee bronnen zwevend zal moeten zijn. Het maken van zwevende spanningsbronnen is moeilijk, daarom worden van de spanningen meestal eerst stromen gemaakt (bijvoorbeeld d.m.v. een weerstand) die daarna worden opgeteld.
In figuur 46 is het optellen van spanningen en stromen te zien. Figuur 46A toont het optellen van stromen, figuur 46B het optellen van spanningen.



Figuur 46: Het optellen van spanningen en stromen

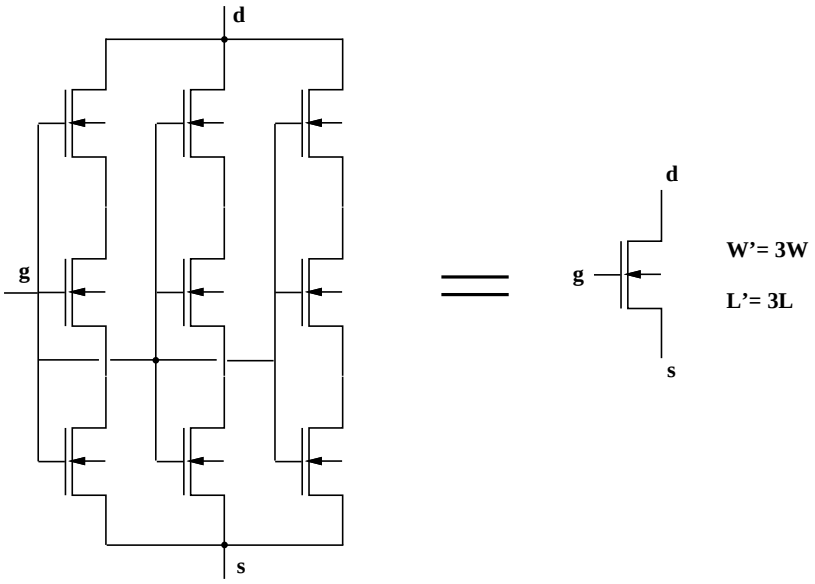
F.3 Bouwstenen voor analoge circuits op de SoG wafer

Er zijn voor de implementatie drie verschillende soorten componenten beschikbaar:

1. MOS-transistoren
2. Weerstanden
3. Condensatoren

MOS-transistoren. Op de sea-of-gates chip is van de PMOS en de NMOS transistor maar één type beschikbaar. Dit zijn transistoren met een zeer kort kanaal. Om toch iets aan schaling van W/L verhoudingen te kunnen doen moeten er transistoren parallel en in serie geschakeld worden. Door twee transistoren parallel te schakelen neemt de breedte (W) van de zo ontstane compound-transistor met een factor twee toe. Door er twee in serie te schakelen (gates aan elkaar) ontstaat een twee keer zo lange.

Eén van de nadelige effecten van een kort kanaal is de lage uitgangsimpedantie die de transistor daardoor heeft. Dit is o.a. bijzonder vervelend in stroomspiegels, waarvan veel gebruik gemaakt wordt. Daarom is er een compound-transistor van 3×3 transistoren gemaakt die bij de ontwerpen zal worden gebruikt, zie figuur 47. In de bibliotheek staan deze compound-transistoren onder de naam `ln3x3` (nmos) en `lp3x3` (pmos).



Figuur 47: De compound-transistor.

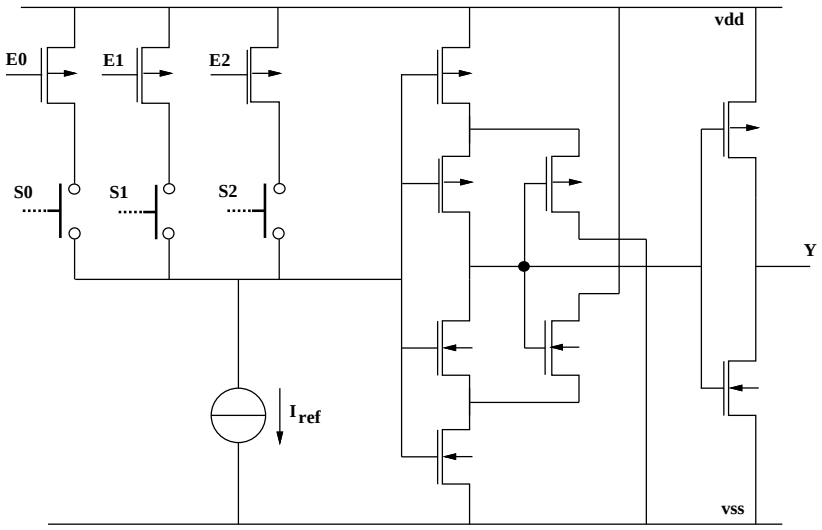
Weerstanden Het is mogelijk de gate van de mos-transistoren te gebruiken als weerstand. Elke gate heeft twee aansluitingen waartussen zich een stukje poly-silicium bevindt dat de gate vormt. De weerstand tussen de twee aansluitingen wordt bepaald door de weerstand van dit stukje poly-silicium en ligt rond de 700 Ω. Door deze weerstandjes in serie of parallel te schakelen kunnen weerstanden van verschillende waarden gemaakt worden. De source en de drain van de betreffende transistoren worden dan niet gebruikt. De “gate-weerstandjes” matchen goed zodat op deze manier nauwkeurig geschaalde weerstanden kunnen worden gemaakt.

Capaciteiten Er zijn verschillende manieren om op de Sea-of-Gates chip capaciteiten te maken. Als er een ”zwevende” (die niet met een aansluiting aan de `vss` verbonden is) capaciteit gemaakt moet worden, kan gebruikt gemaakt worden van de capaciteit tussen de eerste en tweede metaallaag. Capaciteiten die met één poot aan `vss` liggen, kunnen gemaakt worden door verschillende onderliggende lagen met elkaar te verbinden, zodat een soort ”sandwich” ontstaat. De waarden van deze capaciteiten per oppervlakte staan gegeven in appendix B. Omdat op deze manier slechts capaciteitswaarden tot enkele pF’s te maken zijn, zal het in de praktijk vaak voorkomen dat capaciteiten extern aangesloten moeten worden. Hierdoor is het tevens mogelijk om achteraf de waarde ervan nog aan te passen.

F.4 Voorbeelden van implementaties

F.4.1 Toetsenbord-uitlezing

In deze paragraaf zal niet diep worden ingegaan op het scannen van een toetsenbord, maar alleen op het omzetten van de impedantievariaties in een digitaal signaal. In figuur 48 is een simpel schema gegeven.



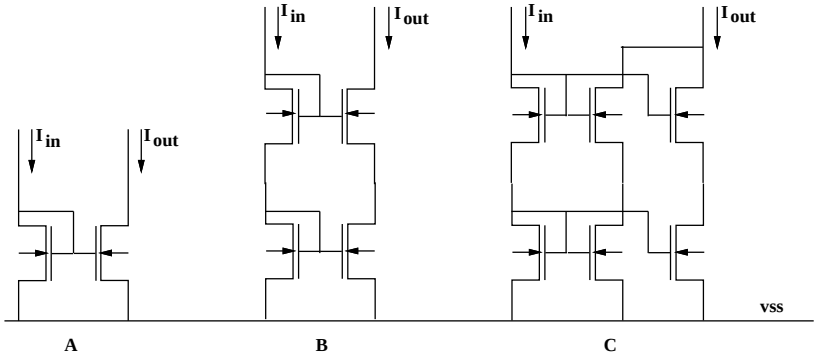
Figuur 48: Methode om een toetsenbord uit te lezen.

Een referentiestroom I_{ref} kan door middel van de ingangen E0, E1 en E2 worden geleid door één van de drie takken met een schakelaar. Het is de bedoeling dat de signalen op E0, E1 en E2 zò worden gekozen dat steeds slechts één weg voor de stroom mogelijk is.
Wanneer I_{ref} goed wordt gekozen, is het mogelijk om de spanningsvariaties als gevolg van het indrukken van de toetsen zodanig te krijgen, dat het signaal direct kan worden aangeboden aan een digitale schakeling. Omdat de flank van dit signaal meestal niet al te steil zal zijn, is het aan te raden om het digitale circuit een zgn. Schmitt-trigger ingang te geven. Deze schakeling is ook weergegeven in de figuur. Een Schmitt-trigger is een schakeling die vanwege meekoppeling een zeer abrupte input-output-karakteristiek krijgt. Overigens hebben de inputbuffers op de chip alle een Schmitt-trigger-ingang.

F.4.2 DA-converter

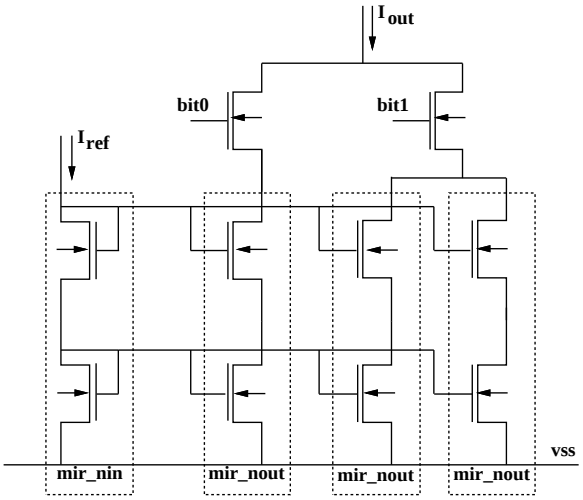
Het principe van de DA-converter wordt behandeld in [4]. Hier staat ook een voorbeeld van een DA-converter op basis van een ladder-verzwakker-netwerk. Andere methoden zijn bijvoorbeeld met behulp van ladingsdeling of geschaalde stroomspiegels. In deze paragraaf zal dieper worden ingegaan op een DA-converter gebaseerd op geschaalde stroomspiegels.
In figuur 49A is het schema gegeven van een eenvoudige stroomspiegel.

Bij een stroomspiegel is het de bedoeling dat uitgaande stroom gelijk is aan de ingaande. Eén van de oorzaken waarom dit nooit precies zo zal zijn, is de eindige uitgangsimpedantie van de uitgangstransistor. Deze gedraagt zich hierdoor niet als een ideale stroombron.
Als dus de drain-spanningen niet hetzelfde zijn, zal I_{out} niet precies gelijk zijn aan I_{in}.
Bij de transistoren die op de Sea-of-Gates chip voorhanden zijn, kan de fout bij een verschil van één Volt al vele procenten bedragen.
Om dit effect te verkleinen kan voor een configuratie als in figuur 49B gekozen worden. De stroomspiegel is nu gecascodeerd. Hierdoor worden de drain-spanningen (van de onderste transistoren) beter aan elkaar gelijk gehouden. Meer informatie over stroomspiegels kunt u vinden in [4].



Figuur 49: A: Een eenvoudige stroomspiegel, B: Een gecascodeerde stroomspiegel, C: Een schalende (gecascodeerde) stroomspiegel

Het is ook mogelijk om met een stroomspiegel stromen te schalen. Door in figuur 49B de uitgangstransistoren dubbel te nemen, zal de stroom I_{out} twee keer zo groot zijn als I_{in}, zie figuur 49C. Op deze manier kunnen dus uit één referentiestroom stromen gemaakt worden die een geheel veelvoud hiervan zijn. Met deze techniek is het heel eenvoudig een DA-converter samen te stellen.

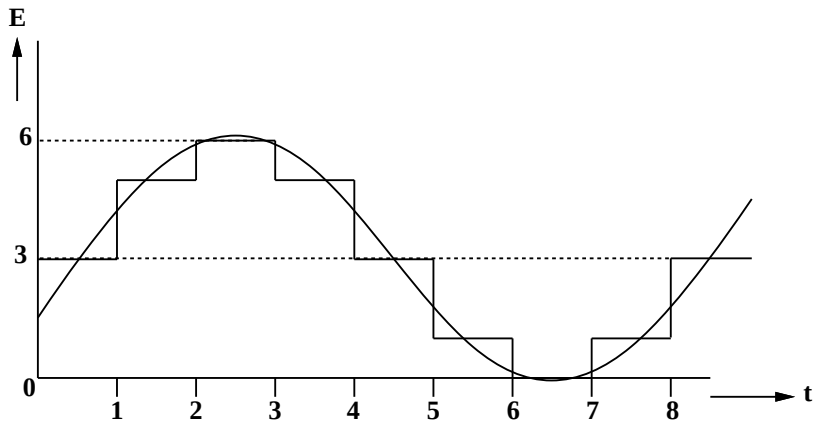


Figuur 50: Een 2-bits DA-converter gebouwd met stroomspiegels en stroomschakelaars.

In figuur 50 is een uitvoering van zo'n DA-converter gegeven.
Met de (digitale) signalen bit0 en bit1 kunnen stromen aan- en afgeschakeld worden. Als bijvoorbeeld bit0 "hoog" is (5 V) en bit1 "laag", zal de uitgangsstroom I_{out} gelijk zijn aan I_{ref}. Wanneer beide signalen hoog gemaakt worden, zal I_{out} $3I_{ref}$ bedragen.
Vanwege het al eerder geschetste probleem met de (te) lage uitgangsimpedantie, is in dit voorbeeld gekozen voor de gecascodeerde stroomspiegel. De basiscellen `mir_nin` en `mir_nout` komen uit de bibliotheek.

F.4.3 Sinus-generatie met een DAC

Eén van de mogelijkheden om een sinus te genereren is een DAC aan te sturen vanuit een digitaal circuit.
Stel dat we een sinus willen genereren in 8 stapjes met een 3-bits DAC. We krijgen dan een signaal dat er uitziet als in figuur 51.



Figuur 51: Benadering van een sinus met een DAC

In tabel 7 staan de signalen die toegevoerd moeten worden om deze ”sinus” te krijgen.

Tabel 7: Signalen voor de DAC

t	E	bit2	bit1	bit0
0	3	0	1	1
1	5	1	0	1
2	6	1	1	0
3	5	1	0	1
4	3	0	1	1
5	1	0	0	1
6	0	0	0	0
7	1	0	0	1

Het is vrij lastig om de distorsie van dit signaal met de hand te berekenen. Programma’s als SPICE doen dit via een Fourier-analyse. Uit simulaties met dit programma blijkt dat een sinusgeneratie in 16 stapjes met een 4-bits DA-converter een harmonische distorsie geeft die kleiner is dan -20 dB (Wanneer de DAC niet-lineair gemaakt wordt, kan bij juiste dimensionering ook volstaan worden met 3-bits).

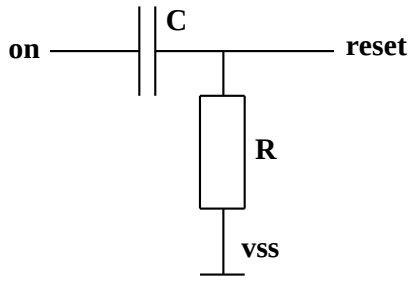
F.4.4 Eindversterker

De luidspreker heeft een impedantie van ongeveer 50 Ω. Om hierin voldoende vermogen te kunnen ontwikkelen is er een eindversterker nodig.

Er kan gebruikt gemaakt worden van een geschaalde stroomspiegel zoals dit ook al is gedaan bij de DA-converters. De schaalfactor wordt zò gekozen dat er voldoende stroom door de luidspreker zal lopen. Het zal duidelijk zijn dat er op deze manier geen “high-performance” versterker ontstaat, maar op dit moment zal de schakeling kunnen voldoen.

F.4.5 Power on reset

Bij geheugenelementen is het vaak niet te voorspellen in welke toestand de schakeling komt wanneer de voedingsspanning aangesloten wordt. Omdat dit in veel gevallen ongewenst is, wordt vaak een zgn. ”power-on-reset” aan de schakeling toegevoegd. Dit schakelingetje genereert een reset-puls op het moment dat de voedingsspanning aangesloten wordt. Hierdoor komt de schakeling altijd in een vaste begintoestand als hij aangezet wordt.
Een mogelijkheid die men vaak ziet, is die met een simpel RC-filtertje, zoals in figuur 52.



Figuur 52: Simpele implementatie power on reset.

Deze schakeling wordt volledig extern aangebracht.
Op dit moment wordt er gewerkt aan een bibliotheekcel die hetzelfde doet. Deze is echter nog niet voor het practicum geschikt.

G Testing SOG chips with the Logic Analyzer

G.1 Introduction

In this document we describe how a Sea-Of-Gates chip in a DIL40 package can be tested with the logic analyzer LA-5580. The purpose is to verify if, given a set of input patterns, the tested device produces the same output patterns as a set of reference output patterns.

A logic analyzer is an instrument that is similar to an oscilloscope: both instruments measure output signals (voltage) as a function of time after some trigger moment. However, important difference are:

- A logic analyzer usually only measures two different signal levels: high and low.
- A logic analyzer is capable of measuring a large number of output signals (e.g. 80).
- Besides measuring output signals, a logic analyzer can simultaneously apply input signals.

The logic analyzer LA-5580 is a hardware box that is controlled (via a USB connection) by a graphical interface program LA5000 that is running a PC. A picture of the interface is shown in Figure 53

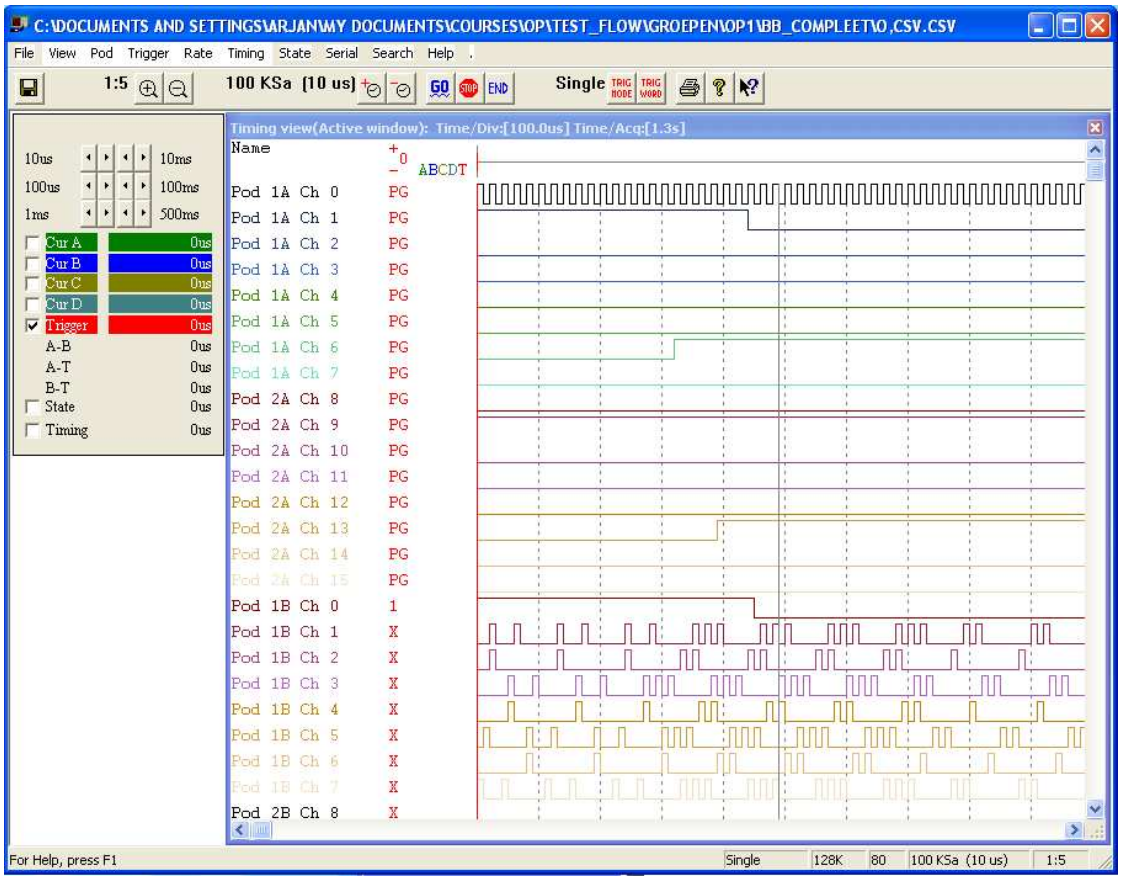


Figure 53: Graphical Interface for the Logic Analyzer LA-5580

G.2 Overview

As input we need a file with reference input and output patterns (.ref file) and a file with information about how the pins of the design are connected to the pins of IO buffers of the chip (.buf file). Typically, both these files have been produced during the design of the chip. Using the program test_flow, these files are converted into a pattern (.csv) file and an initialization (.ini) file that can be used as input for the logic analyzer program, and a scheme that shows how the pods of the logic analyzer should be connected to the pins of the package of the SOG chip. Then, via the the logic analyzer program LA5000, the logic analyzer is used to send the input patterns to the tested SOG device and to collect the resulting output patterns. Finally, the program test_flow is used to compare the measured output patterns against the reference output patterns.

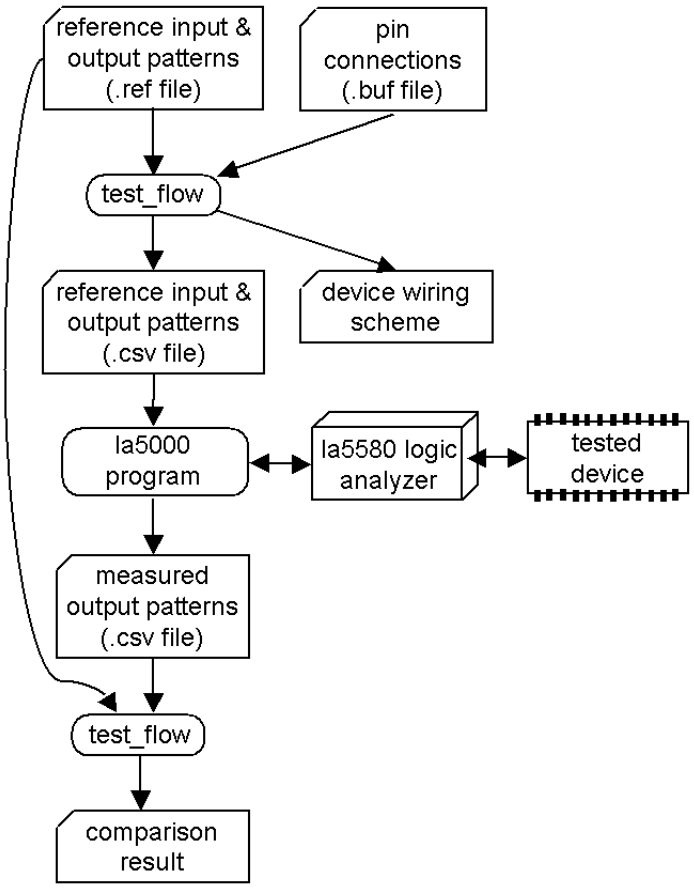


Figure 54: The different steps for testing

G.3 Obtaining the Reference Files

Copy the relevant .ref and .buf files that you have created during the design phase to a writable directory on the local disk of your PC (e.g. C:\wpstmp). Note that it is important to store the files on the local disk because otherwise the logic analyzer software may have problems to read it. Note that

the desktop of the PC is usually not on the local disk in our environment.

G.4 Preparing the Input Data for the Logic Analyzer

Start the program test_flow (T:\ET2805\common\test_flow.tcl). Next click Commands → Read_reffile and open the .ref file. If the same directory contains a .buf file, you will be asked if it is ok to update the information with this file. In that case, click yes. Otherwise click Commands → Update_with_buffile and open the appropriate .buf file. To create a window with input patterns and output patterns in csv format, click Commands → Make_csvfile. In the new csv window, use the command File → Save_csvfile to create the data files that can be used as input for the logic analyzer.

A wiring scheme for how to connect the clips of the wires of the pods to the pins of the SOG device can be obtained by first selecting the appropriate bondbar number for the package in the upper right corner of the test_flow window and next using the command Commands → Show_pod_connections.

G.5 Setting up the Logic Analyzer

Connect the USB cable of the logic analyzer LA-5580 to the upper USB port at the front of your computer. Plug in the pattern generation pods Pod 1A, Pod 2A, etc. in the lower connector rows (label ”board 1”) of the analyzer and the logic (measurement) pods Pod 1B, Pod 2B, etc. in the upper connector rows (label ”board 2”) of the analyzer. Note that the pods have numbers on them. Plug Pod 1 in the leftmost connector, Pod 2 in the second leftmost connector etc.

G.6 Mounting and connecting the SOG chip

Mount the SOG package on the DIL40 connector box. From the separate IC testbox, connect GND (0 Volt) and VDD (5 Volt) to the appropriate connector holes on the DIL40 connector box, using cables with 4mm plugs. Connect the clips of the wires of the pods to the pins of the SOG package according to the device-wiring scheme that was generated using test_flow. Connect at least one black wire of each pod to the GND signal. The pod input Pod 1B(0) at channel 0 will be used as a trigger signal for the acquisition of data and should directly be connected to an output of a pattern generation pod. This connection is given at the bottom of the wiring scheme that was obtained using test_flow.

Switch on the power supply for the SOG chip and switch on the power supply for the logic analyzer. Be careful to only reconnect wires to the pins of the device when the power supply for the SOG chip is switched off.

G.7 Using the Logic Analyzer

Start the Logic Analyzer control program LA5000 Logic Analyzer. If the Timing view window is not yet shown, click Timing → Timing window.

Besides the .csv file that was created by test_flow, also a .ini file with the same base name was created by test_flow. Click File → Open to first open the file .ini file that was created by test_flow.

Next, click View → Clear Data Buffer to clear all signals. Then, click File → Open to open the .csv file that was generated using test_flow. The option Type 2 CSV file must be enabled during this. After reading the .csv file you will see the reference output signals in the Timing window and you may inspect the input signals when you click Pod → Edit Pattern Generator Data (or button EDIT PAT). The reference output signals will be overwritten when capturing analysis data from the logic analyzer.

Then, click Trigger → Go or the Go button to start the analysis. The output of the analysis will appear in the Timing view window.

To prepare for a more exact comparison using the program test_flow, export the output signals using File → Export. For the Data field, select "Group outputs" and select Binary, click OK, and then save the file as type "Comma Sep (.CSV)".

G.8 Comparing the Analyzer Output with the Reference Output

In test_flow, click on Commands → Compare. Next, in the compare window, click File → Compare csv and select the .csv file that was created using the Logic Analyzer. After the file has been read, the measured output signals will be compared against the reference output signals, just before every rising clock edge. On each line you will see from left to right: the time, the clock period number, the input signals, the reference output signals and the measured output signals. In case the reference outputs and measured outputs are different, they have a red colour. You can click on a line to list the different output vectors below each other. If you click on a column in this list, the name of the corresponding output terminal will be highlighted at the sub window at the right. You can use commands from the Commands button in the compare window to navigate through the errors.

Default, all output signals and all inout signals will be compared. By disabling the toggle button inout, only output signals will be compared.

G.9 Trouble Shooting

When the output signals are not according to as expected, carefully check all pod connections. Further you can try to lower the sample rate Thirdly, you can try another IC.

G.10 Further Reading

LA 5000 Logic Analyzer Software Manual (see OP Course Documents on blackboard)
Logic Analyzer LA-5000 series, Link Intruments Inc (http://linkinstruments.com)

Referenties

[1] Peter J. Ashenden. *The Student's Guide to VHDL*. Morgan Kaufmann Publishers, 1998.

[2] Randy H. Katz. *Contemporary Logic Design*. Prentice Hall, 1993.

[3] Jan M. Rabaey. *Digital Integrated Circuits, A Design Perspective*. Prentice Hall Electronics and VLSI Series, 1996.

[4] Richard Spencer and Mohammed Ghausi. *Introduction to Electronic Circuit Design*. Prentice Hall, 2003.