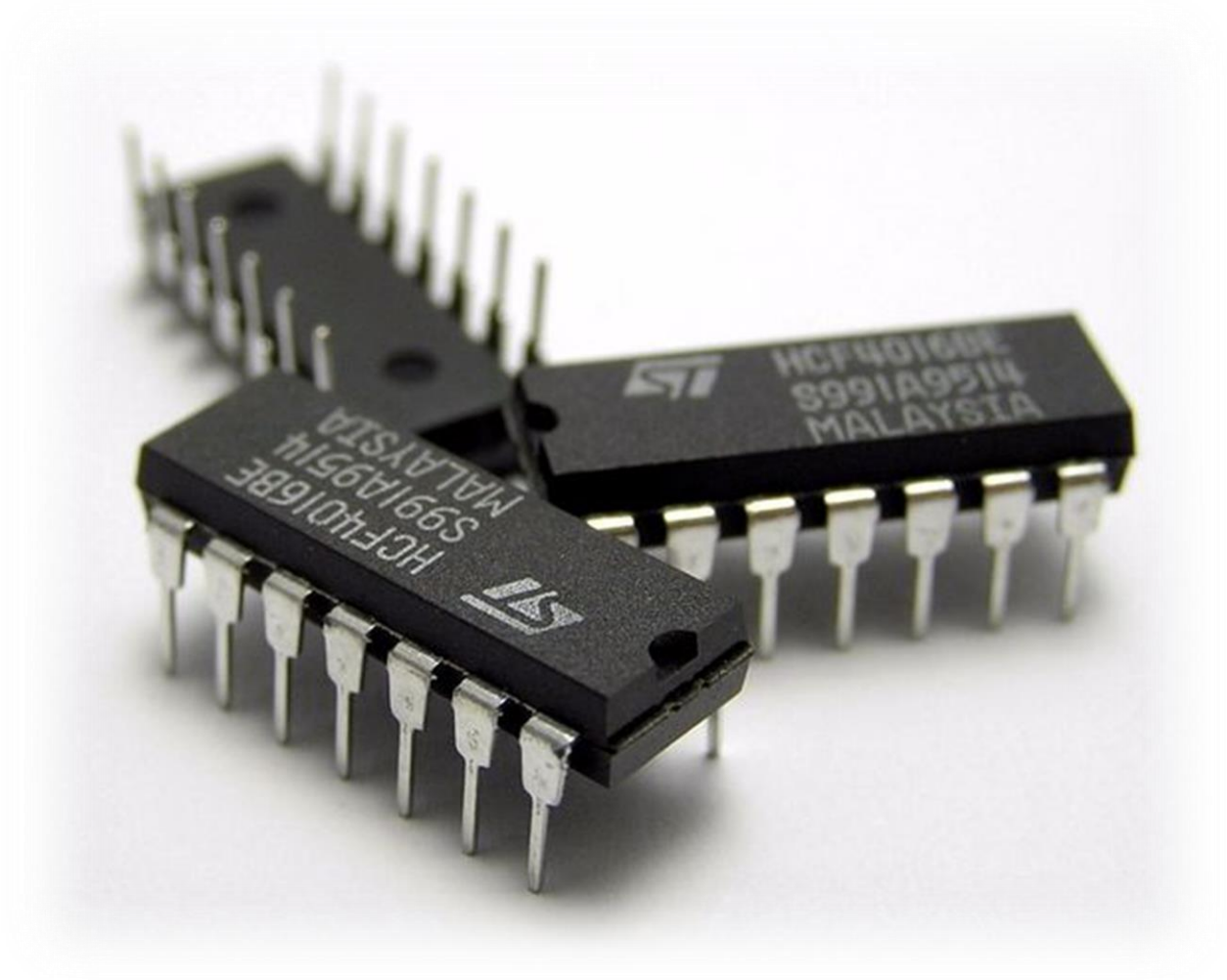
LPA

Life Performance Assessment – Enzo Evers



Fontys
Technology

Contents

Samenvatting LPA

Tijdens dit LPA ga ik leren hoe een PC werkt. Dit ga ik doen door het maken van een 4-bit ALU. Door het maken van de ALU leer ik hoe een encoder en decoder werkt en hoe dit gecombineerd kan worden met een arithmetic unit en logic unit in je PC.

In dit verslag wordt uitgelegd hoe dit werkt, gemaakt wordt en wat ik hiervan geleerd heb.

Samenvatting

Toen ik een video (Bagley, 2014) van Computerphile zag over een bit opslaan met behulp van logic gates bedacht ik me dat je met logic gates eigenlijk een geheel programma zou kunnen maken. Aangezien we voor het LPA een eigen project moeten maken waar we iets van leren bedacht ik mij om een simpele rekenmachine te maken met behulp van logic gates.

De rekenmachine bestaat uit een logic gates die op een breadboard zijn geprikt, ledjes om het antwoord binair te presenteren en switches om een bepaalde bit aan of uit te zetten (om te bepalen met welke getallen gerekend moet worden).

Doelstelling

Ik wil beter begrijpen hoe een computer zijn werk doet. Dit ga ik doen door een versimpelde ALU te maken die twee 4-bit getallen kan optellen en aftrekken, logische operaties op 4-bit getallen kan uitvoeren (AND, OR, XOR).

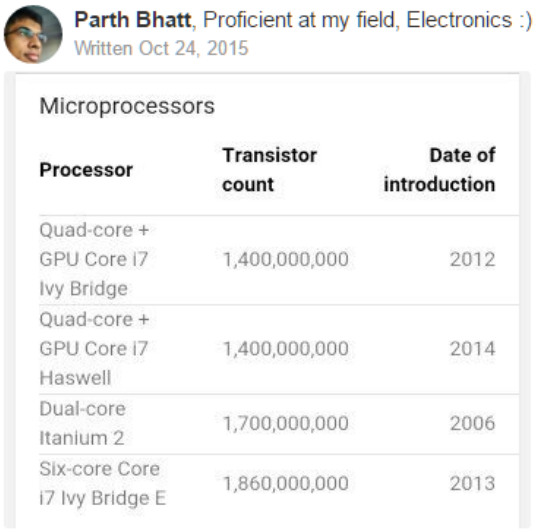
Als dit allemaal is gelukt wil ik ook nog twee 4-bit getallen met elkaar kunnen vermenigvuldigen. Daarbij wil ik ook een 4-bit getal kunnen inverten.

Als dit allemaal is gelukt is en ik heb nog tijd over wil ik twee vier bit getallen met elkaar kunnen delen.

Waarom is dit project belangrijk voor mijn proffesionele ontwikkeling

Door dit project uit te voeren leer ik hoe ik goed moet omgaan met logica en getallen in electronica. Daarnaast leer ik ook (sneller) logische structuren te kunnen begrijpen wanneer ik een circuit diagram zie.

Met logic gates kan je in principe een pc maken. Om te kunnen begrijpen wat er nou eigenlijk echt gaande is in je pc moet je dus wel verstand hebben van logic gates. Wanneer ik dit eenmaal begrijp kan ik ook mijn eigen mini pc maken. Een mini pc maken in de tijd die ik heb tijdens het LPA gaat niet lukken. Daarom hou ik het bij een rekenmachine. Maar wanneer ik snap hoe de rekenmachine werkt kan ik het wel makkelijk uitbreiden naar een mini pc.



Parth Bhatt, Proficient at my field, Electronics :)
Written Oct 24, 2015

Microprocessors

Processor	Transistor count	Date of introduction
Quad-core + GPU Core i7 Ivy Bridge	1,400,000,000	2012
Quad-core + GPU Core i7 Haswell	1,400,000,000	2014
Dual-core Itanium 2	1,700,000,000	2006
Six-core Core i7 Ivy Bridge E	1,860,000,000	2013

As you can see, there are 1.4 billion transistors in a quad core i7. Let's assume on an average 1 gate uses 3transistors, which gives about 450million gates.
PS: this value of gate is based on the average value of the transistors uses per gate, which varies from 2,3,4 or 5 transistors per gate.

612 Views · View Upvotes · Answer requested by Pranav Arora

Figure 1 (Nhatt, 2015)

In de afbeelding links is te zien hoeveel logic gates er eigenlijk in een Processor zitten.

Tijdsplanning

*Zie vernieuwde planning bij de resultaten van dag 2.

<table>
<tr><th>Tijd</th><th colspan="2">Dag 1</th><th colspan="2">Dag 2</th><th colspan="2">Dag 3</th></tr>
<tr><td>09:00</td><td rowspan="2">Onderzoek doen naar het optellen van binaire getallen met logic gates.</td><td rowspan="8">Documenteren</td><td rowspan="2">Onderzoek doen naar het aftrekken van binaire getallen met logic gates.</td><td rowspan="8">Documenteren</td><td rowspan="2">Onderzoek doen naar het vermenigvuldigen van binaire getallen met logic gates.</td><td rowspan="8">Documenteren</td></tr>
<tr><td>10:00</td></tr>
<tr><td>11:00</td><td rowspan="4">Een circuit diagram maken voor de opteller met 2 4-bits getallen.</td><td rowspan="4">Een circuit diagram maken voor de aftrekker met 2 4-bits getallen.</td><td rowspan="4">Een circuit diagram maken voor de vermenigvuldiger met 2 4-bits getallen.</td></tr>
<tr><td>12:00</td></tr>
<tr><td>13:00</td></tr>
<tr><td>14:00</td></tr>
<tr><td>15:00</td><td rowspan="3">Het circuit maken en testen.</td><td rowspan="3">Het circuit maken en testen.</td><td rowspan="3">Het circuit maken en testen.</td></tr>
<tr><td>16:00</td></tr>
<tr><td>17:00</td></tr>
</table>

Vooronderzoek

Wat bestaat er al

Toen ik ging zoeken of dit al eerder was gedaan kwam ik deze (Fourie, 2016) uitgebreide tutorial tegen die precies uitlegt hoe een rekenmachine uit logic gates bestaat (6 video's, 40-60 minuten per stuk).

Een video (Parker, Domino Addition, 2014) van het Youtube kanaal Numberphile illustreert met behulp van domino stenen hoe logic gates werken en hoe je hiermee nummers kunt optellen. In deze video heeft Matt Parker met een team de eerder genoemde video uitgewerkt naar een rekenmachine die bestaat uit 10.000 domino stenen bestaat. (Parker, The 10,000 Domino Computer, 2014).

Figure 2 Figure 1 Matt Parkers domino computer (Parker, The 10,000 Domino Computer, 2014)

Wat ga ik gebruiken

Om dit met electronica te maken heb ik IC's nodig. Wikipedia (Dingley, 2016) heeft een pagina waar alle logic IC's op staan en welk model nummer het is.

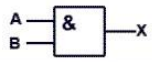
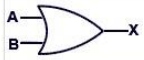
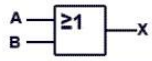
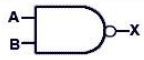
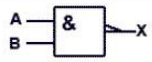
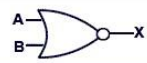
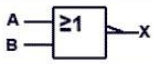
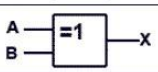
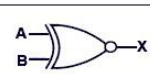
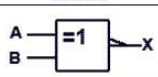
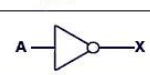
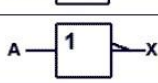
		Table 2.1.1	
ANSI Symbol	**IEC Symbol**	**Description**	**Boolean**
		The AND gate output is at logic 1 when, and only when all its inputs are at logic 1, otherwise the output is at logic 0.	X = A•B
		The OR gate output is at logic 1 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 0.	X = A+B
		The NAND Gate output is at logic 0 when, and only when all its inputs are at logic 1, otherwise the output is at logic 1.	$X = \overline{A \cdot B}$
		The NOR gate output is at logic 0 when one or more of its inputs are at logic 1. If all the inputs are at logic 0, the output is at logic 1.	$X = \overline{A+B}$
		The XOR gate output is at logic 1 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 0.	$X = A \oplus B$
		The XNOR gate output is at logic 0 when one and ONLY ONE of its inputs is at logic 1. Otherwise the output is logic 1. (It is similar to the XOR gate, but its output is inverted).	$X = \overline{A \oplus B}$
		The NOT gate output is at logic 0 when its only input is at logic 1, and at logic 1 when its only input is at logic 0. For this reason it is often called an INVERTER.	$X = \overline{A}$

Figure 3 2 De logic gates en hun functie (Coates, 2016)

Voordat ik de eerste keer aan het LPA ga werken heb ik al een aantal logic gates besteld bij Conrad zodat ik meteen te werk kan gaan.

In het geval dat ik niet (op tijd) aan de juiste componenten kan komen is het altijd nog deze (Temmerman, n.d.) site waar ik de logic gates mee kan simuleren.

Plus

Om daadwerkelijk getallen op te kunnen tellen moet ik gebruik gaan maken van zogenaamde ‘half adders’ en ‘full adders’. De site (4008, n.d.) waar ik onderstaande afbeelding van heb gehaald heeft ook een uitwerking hoe dit in het echt toegepast kan worden.

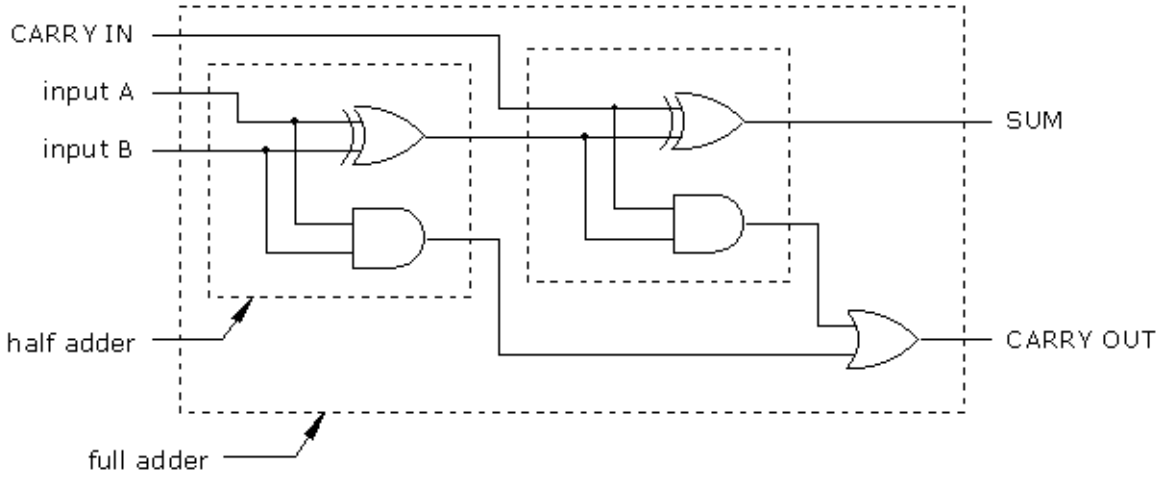


Figure 4 Een digram hoe een half adder en full adder in elkaar zitten (4008, n.d.)

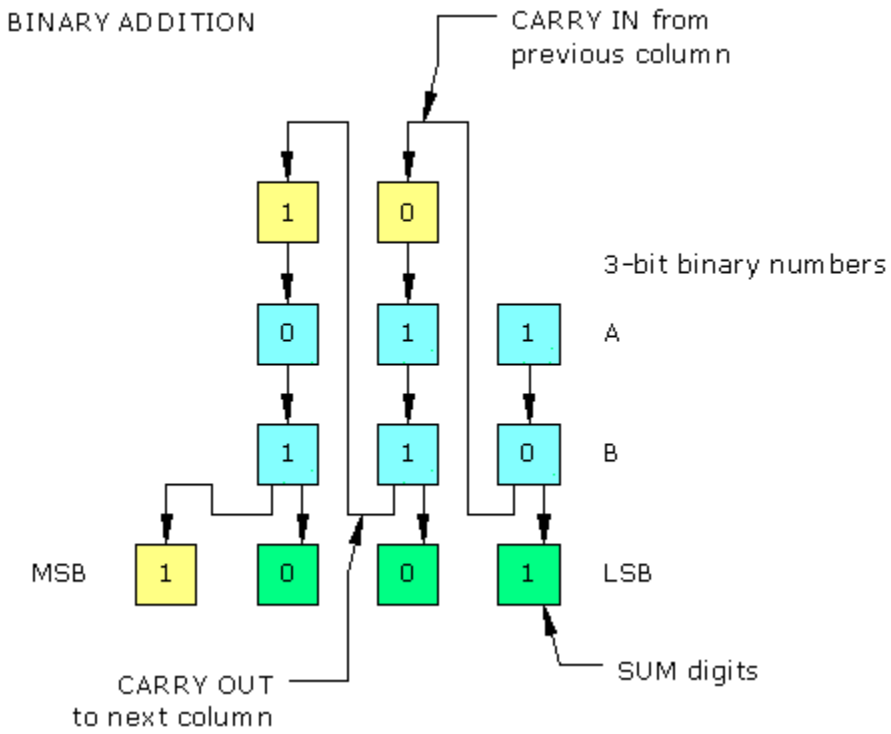


Figure 5 Full adder example (4008, n.d.)

In figure 4 worden de getalen A: 3 (011) en B: 6 (110) opgeteld. Hier zou 9 (1001) uit moeten komen. Dit is ook wat je in figure 4 ziet. Net zoals we in ons tientallig stelsel de tientallen overbrengen naar het volgende getal wordt dat in binaire gedaan met de eenen.

Ingekochte componenten

- **Texas Instruments CD74HC08E Logic IC - Gate AND-Gate 74HC PDIP-14** x20
- **Texas Instruments SN74HC32N Logic IC - Gate OR-Gate 74HC PDIP-14** x20
- **Texas Instruments CD74HCT86E Logic IC - Gate and Inverter XOR (exclusive OR) 74HCT PDIP-14** x20
- **Texas Instruments SN74HCT04N Logic IC - Inverter Inverter 74HCT PDIP-14** x10
- **Texas Instruments 74HC00N Logic IC - Gate and Inverter NAND-Gate 74HC PDIP-14** x20
- **Texas Instruments 74HCT02N Logic IC - Gate and Inverter NOR-Gate 74HCT PDIP-14** x20
- **Texas Instruments 4077 Logic IC - Gate XNOR (exclusive NOR) 4000B PDIP-14** x10
- **Bipolaire standaard vermogenstransistor Diotec BC548C NPN Soort behuizing TO-92 I(C) 200 mA** x20

Dit zijn de links naar de pruduct pagina op Conrad.nl. Op deze pagina staan de specificaties van het product. Op dezelfde pagina staat ook een link naar de datasheet van het product.

Ik zal waarschijnlijk niet alle gekochte producten gaan gebruiken. Maar voor het geval er een IC kapot gaat heb ik in ieder geval vervanging. Daarnaast kan ik het ook nog gebruiken voor eigen gebruik.

Dag 1 (3/11/2016)

Het maken van de 4-bit adder

Onderzoek

Voorafgaand aan dit project heb ik al snel onderzocht hoe het optellen van binaire nummers gaat met behulp van logic gates (zie het kopje Plus). Tijdens mijn onderzoek kwam ik ook te termen *two's complement* en *one's complement* tegen (Wikipedia, 2016). Deze worden echter gebruik tijdens het maken van een subtracter en zal mijn focus daar nu die niet op leggen.

Half adder

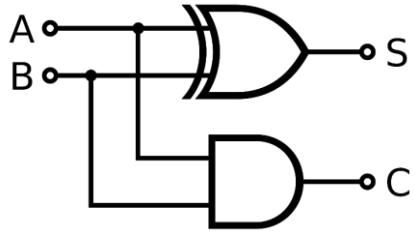


Figure 6 (Wikipedia, 2016)

De bovenstaande afbeelding is een XOR gate (waar S uit komt) en een AND gate (waar C uit komt) te zien. De vergelijking van een de formule voor een XOR gate is $S = A \oplus B$. De 'circle-plus' zoals het heeft heeft de functie van een opteller-modulo-2. Thruth table an een XOR is als volgt.

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Figure 7 (Administrator, 2015)

De half adder zorgt er eigenlijk voor dat er een overflow, de carry-out (C), overgebracht kan worden naar de volgende full adder. Op deze manier kan er een getal van meerdere bits opgeteld worden. De sum, S, is de uitkomt van de huidige bit.

Resultaat

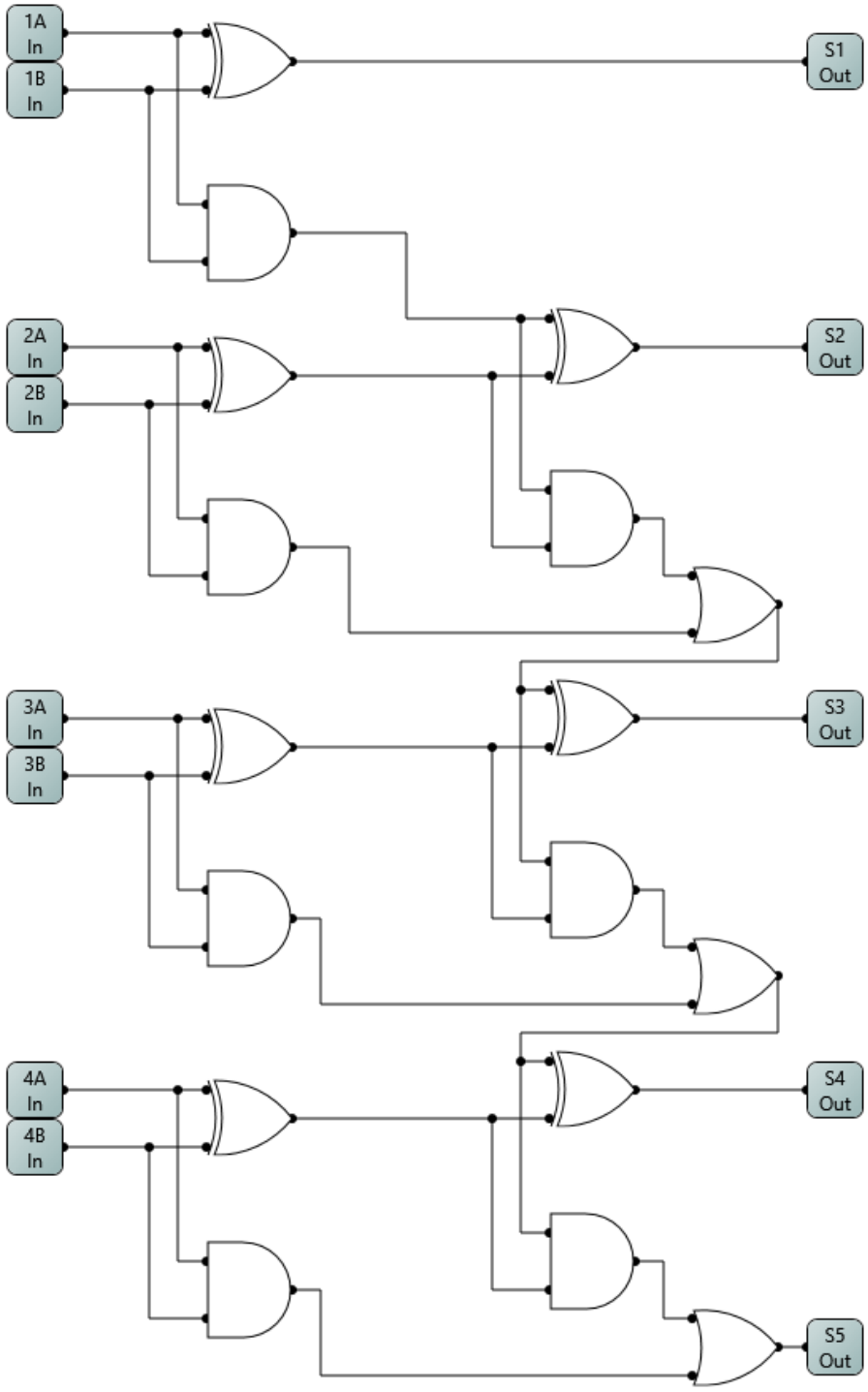


Figure 8 Gemaakt met de Logic Circuit software.

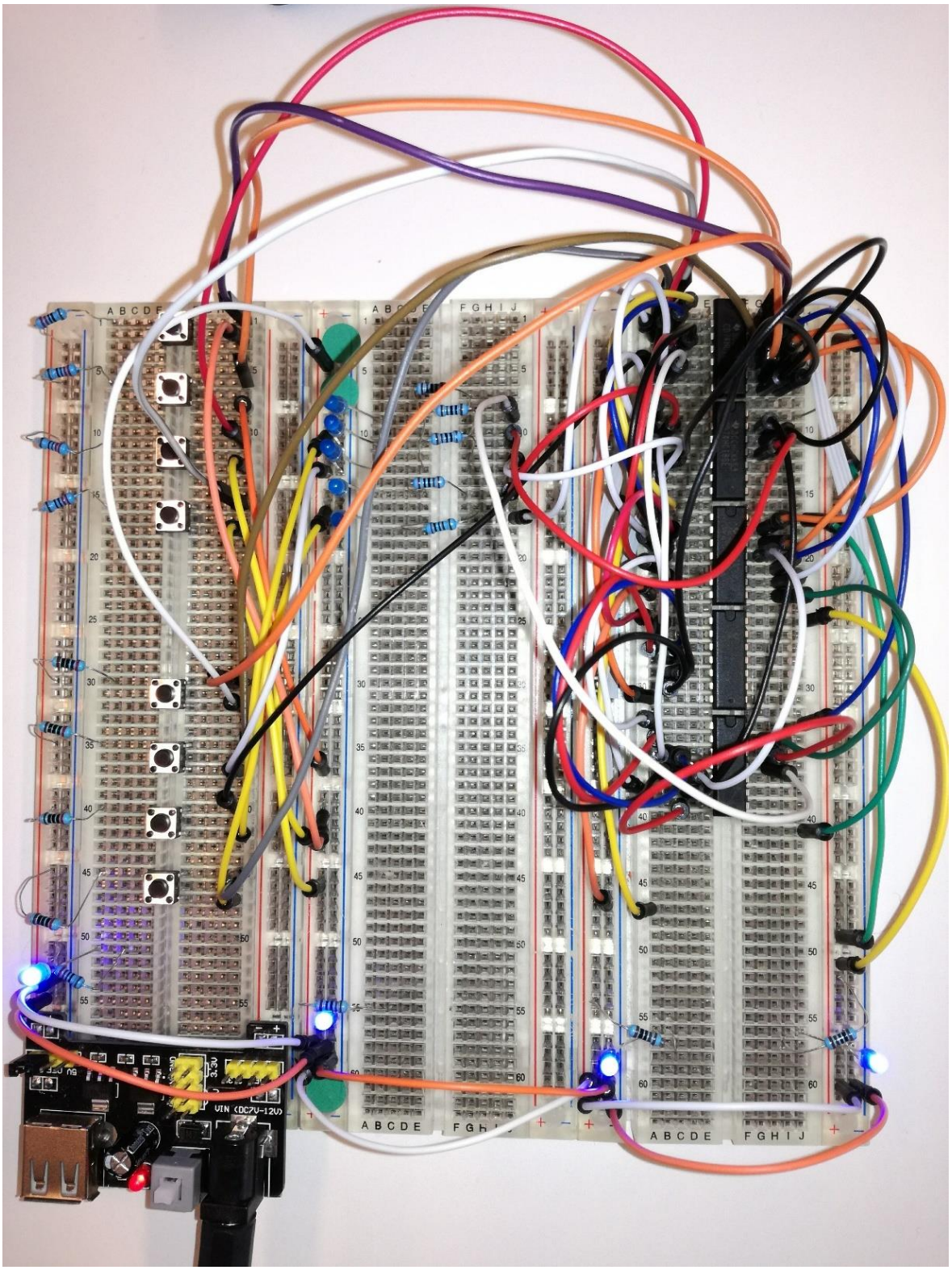


Figure 9 4 bit adder

In Figure 9 4 bit adder is mijn resultaat van de 4 bit adder te zien. Ik heb van te voren het schema gemaakt wat in figuur 4 te zien is. Op het linkse breadboard zijn twee keer vier zien knoppen te zien. Elke setje knoppen moet een 4 bit getal voorstellen. De 5 ledjes op het middelste breadboard laten de uitkomst zien in binair. Van boven naar benenden zijn op het rechtse breadboard te zien 2x 4x XOR gates, 2x 4x AND gates, 1x 4x OR gates.

Problemen en oplossingen

In het begin van de dag heb ik wat tijd verspild door een domme fout. Ik zat met de gedachte dat ik maar vier knoppen nodig zou hebben. In mijn gedachte had ik namelijk dat ik de som van de ingedrukte knoppen (de vier bit getallen) door de ledjes gezien zou laten worden. Eerst zat ik namelijk te denken dat ik overal een van de inputs aan ground zou moeten koppelen (wat natuurlijk onzin is). Toen ik eenmaal een schema. Nadat ik een opnieuw een schema had gemaakt besefte ik opeens dat ik dom aan het nadenken was en dat ik simpelweg twee keer een vier bit getal in de logic gates zou moeten laten komen.

Het Viel ook wel even tegen met hoeveel tijd het kostte om alles in elkaar te zetten. Ik moest erg goed opletten waar ik zou in het schema was en waar elk kabeltje vandaag kwam.

Vooronderzoek dag 2

Voor dag twee is mijn doel een subtractor te maken. Nou heb ik gelezen dat in een Arithmetic Logic Unit (ALU) een van de inputs bij de adder veranderd wordt naar een negative getal door het signaal te inverten en er dan één bij op te tellen. Dit wil ik ook gaan doen zodat ik niet alles weer helemaal opnieuw op hoef te bouwen.

Reflectie

Ik ben tevreden over de vooruitgang van mijn eerste dag. Zoals eerder al beschreven was had ik enkele probleempjes waarvan één gewoon een domme fout was en de ander een tegenvaller in het inschatten van tijd. Ondanks alles heb ik het wel binnen mijn planning af gekregen. Ik had al wel verwacht dat het niet altijd even soepel zou lopen.

Met de tijd die ik over had heb ik nog even gekeken naar hoe het subtracten van binaire getallen in zijn werk gaat. Met wat ik heb gevonden ga ik in dag twee verder werken. Aangezien ik nu al een basis voor een ALU heb staan (de adder) verwacht ik dat de problemen van dag 1 verholpen zijn. Ik hoef namelijk minder kabels te trekken en ook het domme denk probleempje is verholpen. Deze tijd zal ik echter wel nodig hebben om uit te zoeken hoe ik nou kan switchen tussen plus en min.

Herziening van mijn project na opgedane kennis

Waar mijn LPA project begon met het maken van een 4 bit rekenmachine is mijn doel nu een veranderd. Ik wil namelijk een 8-bit computer maken. Dit is uiteraard niet mogelijk binnen het LPA en ik zal dus alleen een deel hiervan in mijn LPA doen.

Voor het maken van mijn 8-bit computer is de planning het volgende:

> Om te beginnen (1) ga uitzoeken welke componenten/modules er in een computer zitten en wat de funtie van deze modules zijn/de architectuur. (2) Dan ga ik onderzoek doen naar hoe elke module nou precies werkt en (3) hoe zich dit is een logic gate circuit uitdrukt. Wanneer alles is onderzocht en de circuits zijn gemaakt (4) ga ik heb op een breaboard maken om te testen of het ook daadwerkelijk werkt. (5) Om het af te maken ga de hele computer op PCB's solderen.

Mijn doel is om, net als de computers die we dagelijks gebruiken, de architectuur modulair te maken. Hiermee bedoel ik dat bijvoorbeeld een andere CPU aangesloten kan worden, extra opslag toegevoegd kan worden, een toetsenbord kan aangesloten worden, etc.. Het liefst wil ik natuurlijk dat ik het ook met een programeer taal kan programeren.

Zoals je misschien al wel merkt moet ik mijn planning gaan aanpassen. Om toch een project binnen drie dagen te kunne maken wil ik het toch in de trend van een rekenmachine houden. Om precies te zijn wil ik in mijn LPA een ALU maken waarmee ik ook kan vermenigvuldigen en delen.

<table>
<tr><th>Tijd</th><th>Dag 1</th><th>Dag 2</th><th>Dag 3</th></tr>
<tr><td>08:45 - 10:00</td><td rowspan="4">Onderzoek naar full adders.</td><td rowspan="2">Onderzoeken hoe een ALU werkt.</td><td>Onderzoeken hoe je decimaal weergeeft in decimaal</td></tr>
<tr><td>10:00 – 11:00</td><td rowspan="3">Circuit diagram maken.</td></tr>
<tr><td>11:00 – 12:00</td></tr>
<tr><td>12:00 – 13:00</td><td rowspan="2">Onderzoeken hoe een binaire vermenigvuldiger werkt.</td></tr>
<tr><td>13:00 – 14:00</td><td rowspan="2">Cicuit diagram maken.</td><td rowspan="2">Onderzoeken hoe een binaire deler werkt.</td><td rowspan="2">Circuit op breadboard bouwen.</td></tr>
<tr><td>14:00 – 15:00</td></tr>
<tr><td>15:00 – 16:00</td><td>Circuit bouwen en testen.</td></tr>
</table>

Dag 2 (10/11/2016)

Arithmetic Logic Unit (ALU)

Een ALU voert rekenkundige en logische operaties uit en is onmisbaar voor een CPU. De uitkomst wordt opgeslagen in de RAM op een register. De input en output van een ALU komen direct vanaf de BUS. De input van een ALU bestaat uit het volgende:

- Intructie woord. Dit laat weten wat er gedaan moet worden. Dit wordt ook wel "**opcode**" genoemd.
- Één of meer "**operands/data**". Dit zijn de getallen waarmee iets gedaan moet worden.
- Soms zit er ook nog een "**format code**" bij. Deze kan bijvoorbeeld laten weten of het een floating point is of niet.

In een ALU zitten vaak ook ingebouwede registers om uitkomsten tijdelek op te kunnen slaan.

(Margaret Rouse, 2005)

De functies van een ALU zijn (Wikipedia, sd):

- Rekenkundige functies
    - Add
    - Add with carry
    - Subtract
    - Subtract with borrow
    - Negate (Two's complement)
    - Increment
    - Decrement
    - Pass through
- Logische functies
    - AND
    - OR
    - XOR
    - Invert (one's complement)
- Bit shift functies
    - Arithmetic shift
    - Logical shift
    - Rotate through carry

Two's complement

In de video ALU design https://www.youtube.com/watch?v=4qH4unVtJkE&t=4s van Ben Eater (Eater, 2016) Wordt uitgelegd hoe je met XOR gates een getal kan negaten/negatief maken. Om dit mogelijk te maken moet ik wel mijn eerst gemaakt full-adder aanpassen. Bij negaten/two's complement moet je namelijk het getal inverten en daar 1 bij op tellen. Dit kan gebeuren door de half adder te vervangen door een full adder en dan een 1 bij de carry-in mee te geven. Dit resulteert in A + (-B) aangezien hier B word negatief.

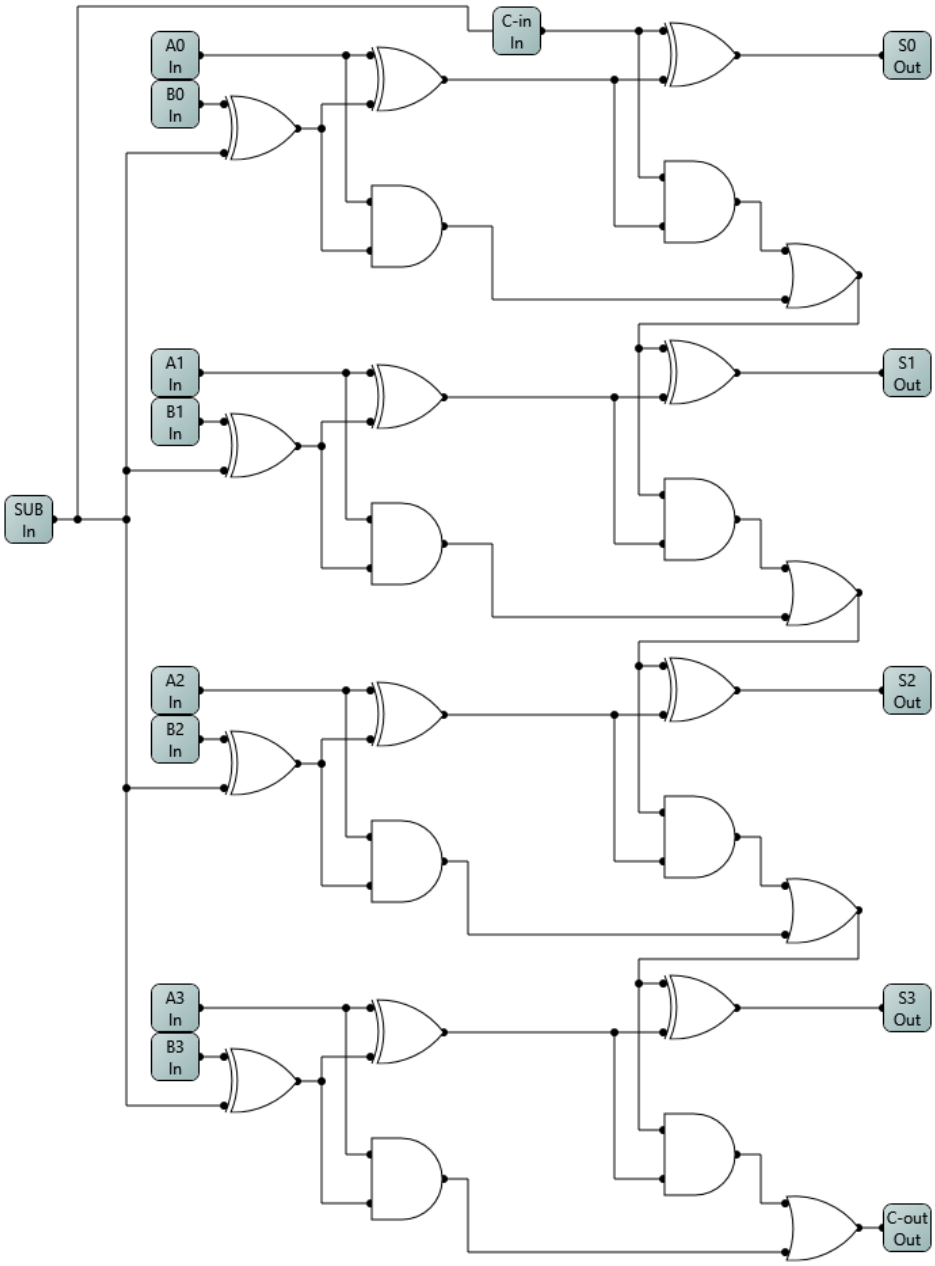


Figure 10 Simple ALU adder and subtractor.

Bit shifting

In dit (Brown, 1999) document wordt uigelegd hoe er met logic gates geshift kan worden. Op deze (Giacomini) site wordt het ook uitgelegd. Bijde sites leggen het net op een iets andere manier uit.

Decoder

Om te kunnen bepalen welke functie er moet worden uitgevoerd moet er gebruik worden gemaakt van een decoder. Deze kan gemaakt worden met AND gates en inverters. Het resultaat van de decoder wordt vervolgens binnen de ALU gebruikt met AND gates om te kiezen wat er moet gebeuren. Het voordeel van een decoder is dat je tot 2^*n* verschillende uitkomsten kan krijgen met *n* signalen.

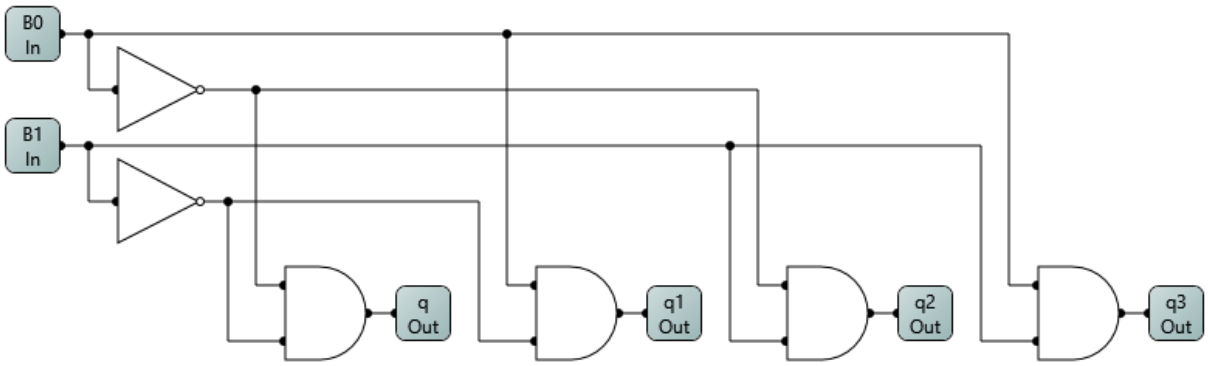


Figure 11 2-bit decode. 4 mogelijke functies met 2 signalen.

Vermenigvuldiger

Ik heb een uitleg (zie hier de dia's) van MIT gevonden over het maken van vermenigvuldigers met logic gates (MIT, 2011). Ik ga gebruik maken van het circuit dan op sheet 12 staat van deze uitleg.

Op deze (Giacomini) site wordt dezelfde manier voor unsigned getallen vermenigvuldigen gebruikt als in de slides van MIT. Hier wordt echter ook een mooi MUL blok gemaakt. Dit maakt het circuit overzichtelijker voor zowel het maken als het lezen van het circuit.

Delen

Op dezelfde site als waar ik bij het vermenigvuldigen waardevolle informatie vandaan haalde staat ook uitgelegd hoe een binaire deler werkt (Giacomini). Om te delen wordt hier gebruik gebruik gemaakt van een full-subtractor. Dit is echter niets anders dan een full-adder waarvan een getal met two's complement is bewerkt. Zoals op de eerder genoemde site al werdt duidelijk gemaakt.

Einde dag 2

Wat ging goed en wat ging slecht

Ik heb over het algemeen mijn planning voor vandaag kunnen afronden. Als ik nu een ALU, vermenigvuldiger en deler zou moeten maken zou dat lukken. Ik heb immers de circuits. Maar dat was niet mijn doel voor vandaag. Mijn doel was om al deze dingen ook te begrijpen. Dit is voor de AlU gelukt, voor de vermenigvulder zo goed als helemaal, en voor de deler een beetje.

Reflectie

Na de eerste dag van het LPA was ik erg enthousiast geworden over het onderwerp en was ik thuis gaan kijken hoe je nou een gehele computer zou kunnen maken. Mijn eigenlijke doel voor dag twee was het maken van een subtractor. Aangezien ik er achter kwam dat dit met een kleine aanpassing aan mijn adder gedaan kon worden wilde ik iets grootsers maken. Nou blijkt dat ik (voor de mogelijke tijd) te hoog had gegrepen. Zoals eerder al gezegd heb ik wat ik nodig heb, maar kwam er bij het begrijpen meer kijken dan ik dacht.

Wanneer ik mijn doelen (deels) niet kan halen voelt dit voor mij in eerste instantie als falen. Maar als ik eens goed nadenk realiseer ik me dat dit niet was gebeurd als ik me gewoon aan mijn eerste planning had gehouden. Ik moet er dus op letten dat ik mijn interesses niet de baas over de planning laat hebben.

Tijdens het gesprek met mijn LPA-tutor is gebleken dat ik een voor dag drie een duidelijk doel moet maken met wat ik wil laten zien aan het einde. Ik moet mijn doel enigszins aanpassenen en een MoSCoW table maken.

Functie product	MoSCoW
Twee 4-bit getallen bij elkaar optellen.	M
Twee 4-bit getallen van elkaar aftrekken.	M
AND op twee 4-bit getallen kunnen toepassen.	M
OR op twee 4-bit getallen kunnen toepassen.	M
XOR op twee 4-bit getallen kunnen toepassen.	M
Een 4-bit getal kunnen inverten	s
Twee 4-bit getallen vermenigvuldigen.	S
Twee 4-bit getallen delen.	C

Omdat mijn doel is het beter begrijpen hoe een PC werkt hoef ik niet per se een vermenigvuldiger en deler te maken. Dit zijn namelijk gewoon meerdere adders en subtractors in een specifieke volgorde. Door een ALU te maken kom ik eigenlijk al meerendeel van de functies binnen pc tegen. Namelijk logic operation, rekenkunde en adresseren (opcode).

Tijd	Dag 3
08:45 – 13:00	Circuit ontwerpen.
13:00 – 16:00	Circuit op een breadboard bouwen en testen.

Dag 3 (26/1/2017)

Encoder

Om een encoder te maken is het handig om een K-map (karnaugh map) te maken. Het youtube kanaal Neso Academy (Neso Academy, 2014) heeft een aantal video's over K-maps. Deze video's heb ik gevolgd. De drie delige serie video's die de basis van de K-map uitleggen hebben mij geholpen met het begrijpen hoe ik het kan gebruiken (de eerste video). De video die over 5 variabel K-maps ging heeft me het meest geholpen met het bouwen van mijn eigen encoder. Met het onderzoek naar de encoder en K-map ben ik thuis al even mee begonnen. Ik wil aan het eind van de dag namelijk een werkende ALU hebben en om een geheel nieuw (abstract) iets te leren, de karnaugh map, en toe te passen dacht ik iets meer tijd nodig te hebben dan een dag.

a	b	c	d	e	f			
+	-	&	|	^	~	e2	e1	e0
0	0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0	1
1	0	0	0	0	0	1	1	0

Figure 12 Truth table voor waar mijn encoder aan moet voldoen.

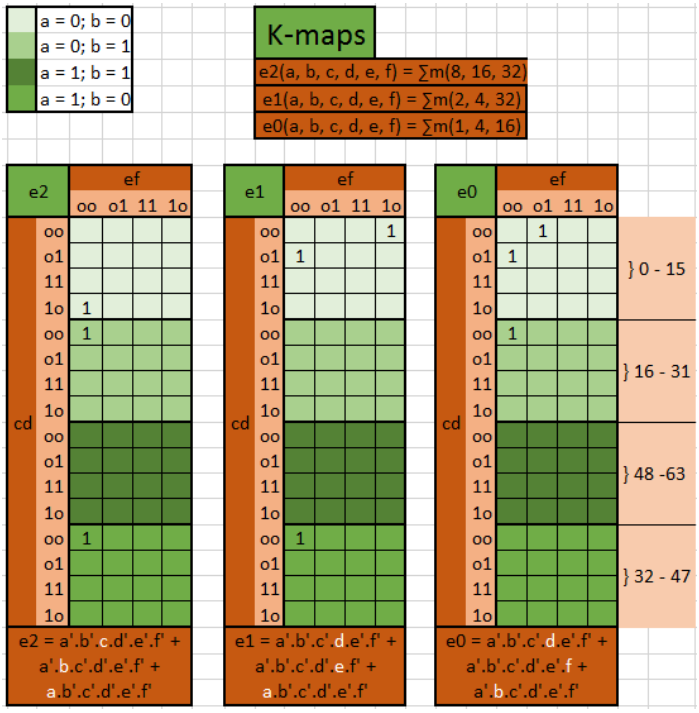


Figure 13 Mijn Karnaugh map (K-map) voor de encoder. Een letter met ' stelt complement voor. Gemaakt in excel.

ALU design

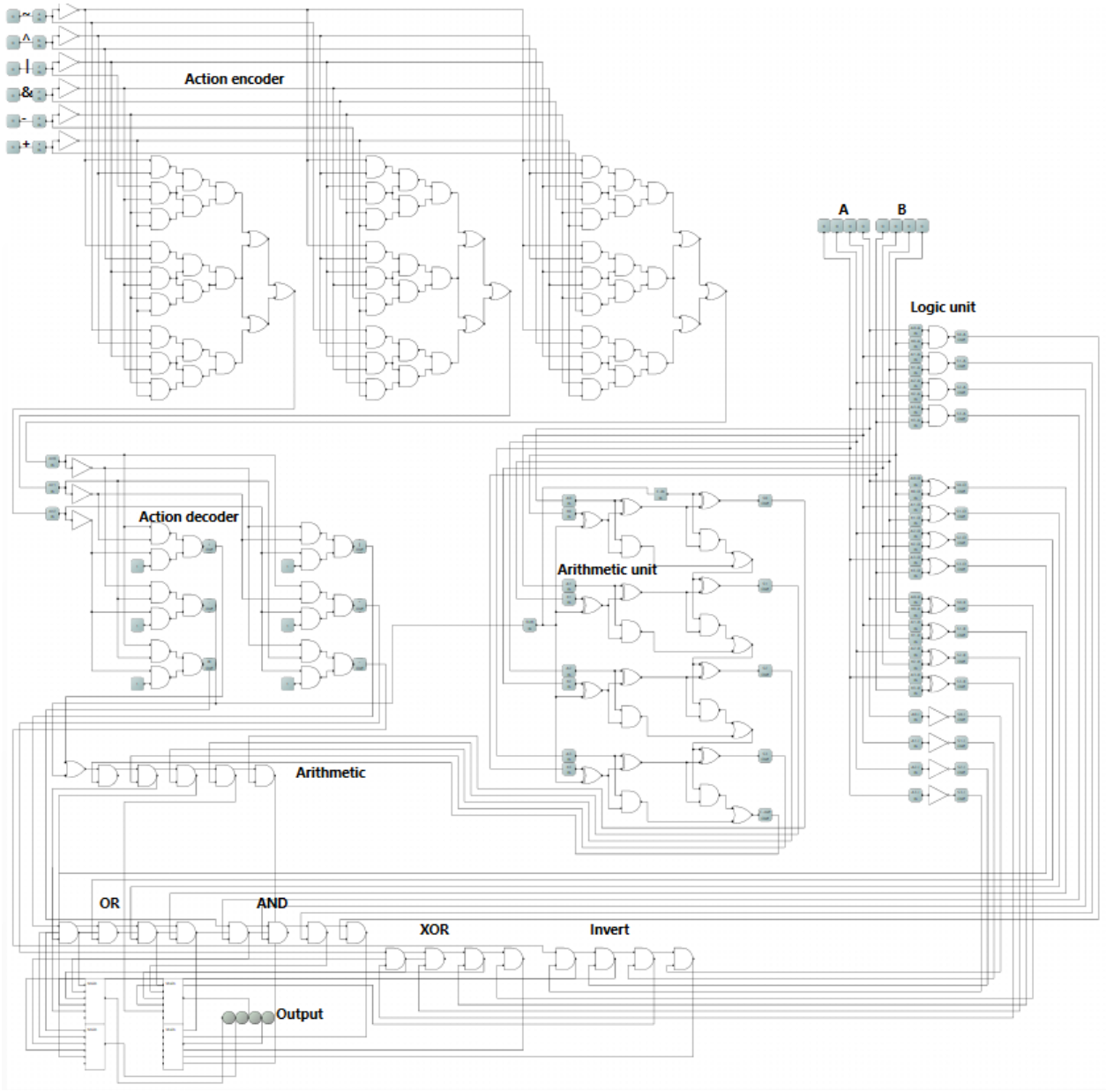


Figure 14 Design voor een 4-bit ALU

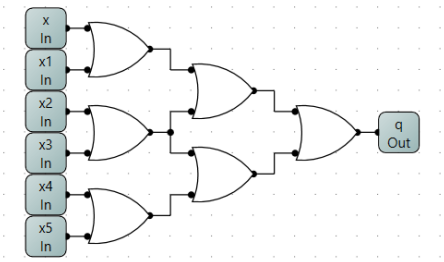


Figure 15 Bit enable constructie.

De 4 rechthoekige blokken links onder bevatten de constructie uit figuur 15. In LogicCircuit heb je niet oneindig veel ruimte voor het circuit en heb ik dus figuur 15 apart moeten maken om ruimte te besparen. Door het gebrek aan ruimte zijn links onder soms ook draden over elkaar waardoor het soms niet helemaal duidelijk is wat waar nou naartoe gaat. Wat de bedoeling is is dat er wordt gekeken of de bepaalde output bit aan moet gaan op basis van de aangezette **action** en de **output van de units**. Elke blokje is voor 1 bit.

Bouwen

Benodigde componenten:

- AND x104 104/4 = 26 IC's (heb er maar 20)
- NOR x18 18/4 = 5 IC's
- OR x42 42/4 = 11 IC's
- XOR x16 16/4 = 4 IC's
- Invert x13 12/6 = 3 IC's

*Alle IC's hebben 4 logic gates behalve invert, die heeft er 6.

Ik heb maar 20 AND gates en kom er dus 6 te kort. Ik heb nog wel 20 NOR gates en van 3 NOR gates kan ik 1 AND gate maken. Voor 6 AND gates heb ik dus 6*3 = 18 NOR gates nodig.

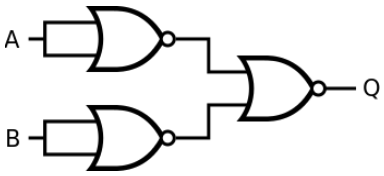


Figure 16 AND gate gemaakt uit NOR gates. (-, 2017)

Op een breadboard kan ik 63(het aantal rijen op een breadboard)/7 = 9 IC's kwijt. Voor alleen de IC's heb ik dus (20(AND) +11(OR) + 4(XOR) + 3(Invert) + 18(NOR) = 56) 56/9 = 7 breadboards nodig. Voor de input en uitkomst heb ik ook nog een breadboard nodig dus in totaal 8 breadboards.

Probleem

Aangezien de tijd die ik heb zie ik het nut van het bouwen niet meer helemaal. Ik heb al een werkende simulatie gemaakt in het programma LogicCircuit. Daarnaast spreek ik uit ervaring van dag 1 dat er altijd wel ergens een draadje verkeerd zit en aangezien ik niet genoeg kabels heb om met kleure code te werken gaat dat nu ook bijna zeker gebeuren. De verkeerde connectie zoeken zal dan veel tijd gaan innemen. Ook heb ik van dag 1 al gemerkt hoe lang het bouwen van alleen een 4-bit adder kost. Het bouwen van de ALU zal minimaal twee keer zo lang duren aangezien de grootte. Ik heb in mijn originele plan ook gezegd dat ik een sumilatie zou kunnen gebruiken om het product te demonstreren.

Einde dag 3

Wat ging goed en wat ging slecht

Wat goed ging is dat ik het doel wat ik heb veranderd op het einde van dag twee gehaald heb. Ik heb geleerd hoe een PC in de basis werkt en dat is wat ik wilde leren. De planning voor dag drie is iets minder gelukt i.v.m. tijd en de hoeveelheid hardware die nodig is. Ik heb het circuit wel kunnen maken als simulatie maar het fysiek bouwen is dus niet gelukt.

Reflectie

Ik vind dat ik de afgelopen drie dagen opzich goed bezig ben geweest. Ik had mijn project kunnen laten zoals het in eerste instantie was, maar dan zou ik niet geleerd hebben wat ik echt zou willen leren. Waar ik iets minder tevreden mee ben is dat ik niet goed heb kunnen inschatten hoeveel tijd het bouwen in beslag neemt. Opzich had ik na dag 1 al kunnen weten dat het bouwen van een ALU op een breadboard in een middag niet zou lukken. Wat ik wel goed vind is dat ik op dag 3 de beslissing heb genomen de ALU niet op het breadboard te maken en me te focussen op de simulatie en documentatie.

Thuis heb ik soms ook iets aan onderzoek gedaan over ALU voor het LPA. Dit was over het algeem uit eigen interesse en niet per se voor het LPA. De K-map was wel duidelijk voor het LPA omdat dit anders te veel tijd in beslag zou nemen.

Bibliography

-. (2017, January 25). *https://en.wikipedia.org/wiki/AND_gate*. Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/AND_gate

4008. (n.d.). Retrieved from http://www.doctronics.co.uk/: http://www.doctronics.co.uk/4008.htm

Administrator. (2015, July 14). *Exclusive OR Gate(XOR-Gate)*. Retrieved from http://www.electronicshub.org/: http://www.electronicshub.org/exclusive-or-gatexor-gate/

All about circuits. (n.d.). *Special-output Gates*. Retrieved from www.allaboutcircuits.com: http://www.allaboutcircuits.com/textbook/digital/chpt-3/special-output-gates/

Bagley, D. S. (2014, June 23). How Computer Memory Works. (S. Riley, Interviewer)

binarymath. (n.d.). *Multiplication & Division.* Retrieved from www.binarymath.info: http://www.binarymath.info/multiplication-division.php

Brown, B. (1999). *Shifting and Shifters.* Retrieved from ksuweb.kennesaw.edu/: http://ksuweb.kennesaw.edu/faculty/rbrow211/papers/shifter.pdf

Coates, E. (2016, March 31). *Digital Logic*. Retrieved from http://www.learnabout-electronics.org/: http://www.learnabout-electronics.org/Digital/dig21.php

Computer organization and architecture. (2014, November 26). Retrieved from http://blogkami1995.blogspot.nl/: http://blogkami1995.blogspot.nl/2014/11/digital-logic.html

Dingley, A. (2016, August 28). *List of 7400 series integrated circuits*. Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits

Eater, B. (2016, September 10). *Building an 8-bit breadboard computer!* Retrieved from www.youtube.com: https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTBypGqImE405J2565dvjafglHU

electronics-tutorials. (n.d.). *Combinational Logic*. Retrieved from www.electronics-tutorials.ws: http://www.electronics-tutorials.ws/category/combination

Fourie, R. (2016 , March - April 12 - 4). *Reon Fourie*. Retrieved from www.youtube.com: https://www.youtube.com/watch?v=qLyIULjAtxY&list=PL80Zqpd23vJcyLmSv7UTFY9nLyZySS2_E&index=1

Giacomini, D. (n.d.). *Chapter 3. Combinational circuits* . Retrieved from http://ba.mirror.garr.it/: http://ba.mirror.garr.it/1/groundup/a36.html

madmaxx. (2012, November 27). *Lets Build 8 Bit Computer*. Retrieved from www.youtube.com: https://www.youtube.com/watch?v=bCVT1BtlZn0&list=PLNUL7DzXzp_J4TztZYOVtayTfp0DV1z5H

Margaret Rouse, D. B. (2005, September). *arithmetic-logic unit (ALU)*. Retrieved from http://whatis.techtarget.com/: http://whatis.techtarget.com/definition/arithmetic-logic-unit-ALU

MIT. (2011). *L09.pdf.* Retrieved from web.mit.edu: http://web.mit.edu/6.111/www/f2011/handouts/L09.pdf

Neso Academy. (2014, July 29). *Neso Academy*. Retrieved from www.youtube.com: https://www.youtube.com/user/nesoacademy/featured

Nhatt, P. (2015, October 24). *How many logic gates are there in a computer?* Retrieved from www.quora.com: https://www.quora.com/How-many-logic-gates-are-there-in-a-computer

Palmer, P. (1997, July 22). *Computer Architectures 5.* Retrieved from http://info.ee.surrey.ac.uk/: http://info.ee.surrey.ac.uk/Teaching/Courses/msccomp/notes/comparch5.html

Parker, M. (2014, April 4). Domino Addition. (B. Haran, Interviewer)

Parker, M. (2014, April 1). The 10,000 Domino Computer. (-, Interviewer)

Randy. (1996, July 14). *5.5 Combinational Multiplier.* Retrieved from www2.elo.utfsm.cl : http://www2.elo.utfsm.cl/~lsb/elo211/aplicaciones/katz/chapter5/chapter05.doc5.html

Temmerman, K. (n.d.). *logic-lab*. Retrieved from www.neuroproductions.be: http://www.neuroproductions.be/logic-lab/

The university of Utah. (2011). *CS/EE 5830/6830 VLSI Architecture.* Retrieved from http://www.eng.utah.edu/: http://www.eng.utah.edu/~cs6830/

University of Pittsburg. (n.d.). Retrieved from http://www.pitt.edu/: http://www.pitt.edu/~kmram/CoE0147/lectures/datapath3.pdf

Wikipedia. (2016, October 25). *Adder (electronics).* Retrieved from https://en.wikipedia.org: https://en.wikipedia.org/wiki/Adder_(electronics)

Wikipedia. (2016, May 25). *File:Half Adder.svg*. Retrieved from https://commons.wikimedia.org: https://commons.wikimedia.org/wiki/File:Half_Adder.svg

Wikipedia. (n.d.). *Arithmetic logic unit*. Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Arithmetic_logic_unit