Godkendelsesopgave 1
Fra NAND til ALU

Maskinarkitektur, blok 1 2011

Introduktion

Dette er den første godkendelsesopgave på DIKUs bachelorkursus *Maskinarkitektur*. Den består af tre delopgaver. Opgaven skal udarbejdes i grupper af 2-3 personer, afvigelser herfra kræver tilladelse fra jeres instruktor.

Programmel

Til at løse opgaverne skal I bruge programmet *Logisim*, som kan hentes fra `http://ozark.hendrix.edu/~burch/logisim/`. Derudover skal I bruge et komponentbibliotek som indeholder MIPS-komponenter, der imiterer dem i lærebogen [1]. Biblioteket er tilgængeligt fra `http://github.com/downloads/andersbll/logisim-diku/logisim-diku-1.0.jar`. Bemærk at Logisim ikke er et professionelt stykke software og at fejl kan opstå i ny og næ. I opfordres derfor til at gemme jeres kredsløb regelmæssigt og foretage backup jævnligt så I ikke mister store arbejdsindsatser hvis uheldet er ude.

Aflevering

Opgaven skal afleveres af en repræsentant fra gruppen på Absalon senest mandag d. 19. september kl 8:00. Der skal afleveres Logisim-filer som løser opgaven samt en mindre rapport (i PDF-format), der dokumenterer jeres arbejde (inddrag gerne billeder af jeres kredsløb). Rapporten kan være på enten dansk eller engelsk.

Om teknisk rapportskrivning

Erfaring fra tidligere år viser at studerende på dette kursus kun har ganske lidt rutine i at skrive tekniske rapporter. Dette er hverken HCI eller Førsteårsprojektet og I skal derfor ikke skrive mere end hvad der er nødvendigt for at kunne dokumentere jeres løsning og de overvejelser I har gjort. En god tommelfingerregel er at hver eneste sætning I skriver skal viderebringe ny information til læseren. Herudover skal I passe på med jeres ordvalg og brug af sproget. Undgå overdramatiseringer og talesprog.

Vi mener at denne opgave burde kunne dokumenteres på ca. 2-3 normalsider, eksklusiv figurer. Bemærk at dette hverken er en hård øvre eller nedre grænse. Derudover anbefales det kraftigt at jeres rapport indeholder følgende

- En introduktion, hvori I kortfattet redegør for hvorvidt opgaven er blevet løst.
- Beskrivelse af jeres løsning, gerne inklusiv en diskussion af forskellige løsningsmuligheder samt argumentering for jeres valg (hvis I har foretaget nogle).
- Redegørelse for korrektheden af jeres løsning. Dette kan foregå enten ved at argumentere for løsningen eller ved at dokumentere en succesfuld afprøvning.

Hjælp til opgaven

Alle spørgsmål vedrørende opgaven bedes stilles på kursets forum på Absalon for at undgå forfordeling mellem kursusdeltagerne. Husk at benytte jer af øvelsestimerne, hvor opgaven vil blive gennemgået.

G1.1 AND, OR, NOT, XOR

Konstruer komponenterne AND, OR, NOT og XOR kun ved brug af NAND-gates. NAND-komponenterne må kun tage to indgange af 1 bits bredde. Konstruer alle komponenter i en enkelt Logisim-fil og opret et *subcircuit* per komponent.

G1.2 4-bits ALU

Konstruer en 4-bits *Aritmetisk Logisk Enhed* (ALU) der benytter sig af *to-komplement* talrepræsentationen. ALUen skal konstrues fra bunden og det er derfor kun tilladt at benytte sig af Logisim-komponenterne fra *Base*, *Wiring* og *Gates*-bibliotekerne.

Jeres ALU skal have de samme ind- og udgange som i lærebogens [1] figur C.5.14 (bemærk at *CarryOut*-udgangen ikke skal være der) og forstå operationerne i tabellen på figur C.5.13. Det er selvfølgelig meningen at I skal hente inspiration fra de andre figurer i Appendix C.

Udleveret skelet

For at ensrette jeres kredsløb, så de er nemmere for instruktorerne at rette, skal I arbejde videre på det udleverede ALU-skelet. Jeres opgave er dermed at udfylde indholdet af ALU-komponenten så den opfører sig som forventet. Som en hjælp til afprøvningen er der blevet tilføjet et semiautomatisk testkredsløb, hvor opførslen af jeres ALU sammenlignes med en korrekt fungerende *black box* ALU.

Råd og vink

- Husk at modularisere designet af jeres kredsløb. Benyt jer af *subcircuits* for at undgå redundant arbejde og mindske kompleksiteten af kredsløbet.
- Gør en indsats for at layoute jeres kredsløb på overskuelig vis samt at navngive jeres komponentindgange og -udgange. Det vil betale sig i det lange løb, når I skal finde fejl i kredsløbet.
- Benyt jer af muligheden for at oprette kredsløb ud fra logiske udtryk (menupunktet *Combinational Analysis*).
- Vi anbefaler følgende fremgangsmåde
  1. Lav 1-bit full adder (Figur C.5.2)
  2. Lav 1-bit ALU (Figur C.5.10)
  3. Lav 4-bit ALU (Som Figur C.5.12)
- Bogen angiver ikke præcist hvordan 4-bits ALUen kan detektere overløb. Benyt eventuelt tabellen på Figur 3.2, side 226 som hjælp.

G1.3 32-bits skifteenhed

Konstruer en skifteenhed (engelsk: *shift unit*) der understøtter operationerne

- `<<` (*Shift left logical*, `sll`) svarende til venstreskift i C.
- `>>` (*Shift right logical*, `srl`) svarende til højreskift af en `unsigned int` i C.
- `>>>` (*Shift right arithmetic*, `sra`) svarende til højreskift af en `int` i C.

Dvs. forskellen på `srl` og `sra`, er at `srl` skifter 0-bits ind i mest betydende bit, mens `sra` bibeholder det eksisterende fortegn i mest betydende bit, og skifter det ind i de mindre betydende bits. For eksempel gælder (for 4-bits tal) 1011 << 1 = 0110, 1011 >> 1 = 0101 og 1011 >>> 1 = 1101

Skiftekomponenten har følgende ind- og udgange.

`in`	32-bits indgang	
`shamt`	5-bits indgang	Forkortelse af *shift amount*. Talværdien (tolket som unsigned) af `shamt` angiver hvor mange pladser `in` skal skiftes.
`result`	32-bits udgang	
`op`	2-bits indgang	Kontrolsignal der indikerer 00 = `sll` 01 = udefineret 10 = `srl` 11 = `sra`

Til at løse opgaven er det kun tilladt at benytte sig af Logisim-komponenterne fra *Base*, *Wiring* og *Gates*-bibliotekerne samt *Multiplexer*-komponenten fra *Plexers*.

Udleveret skelet

Ligesom i G1.2 får I udleveret et simpelt skelet, som I skal benytte til at konstruere skifteenheden. Denne gang er der ikke nogen hjælp til afprøvning af komponenten.

Råd og vink

- Konstruer skiftekomponenten som en serie af fem multiplexere, der skifter `in` henholdsvis $2^n, n = 0, \ldots, 4$ pladser (men kun hvor de tilsvarende bits i `shamt` er sat).

Litteratur

[1] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers, 4th edition, 2008.