G1
Maskinarkitektur
Efterår 2011

Jens Fredskov
Naja Mottelson (vsj465)
Søren Pilgård (vpb984)

18. september 2011

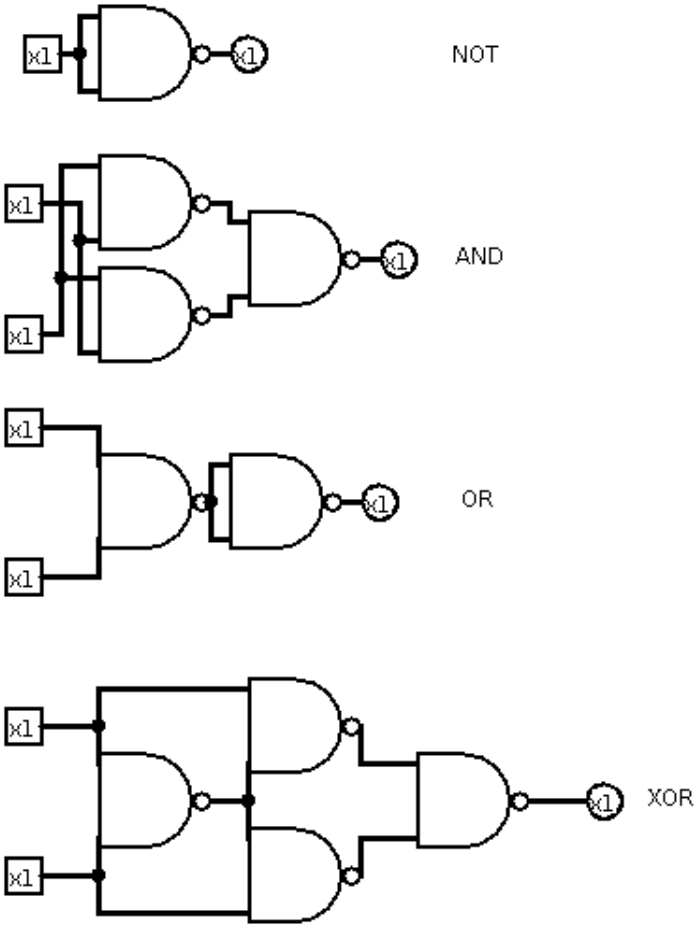
Indhold

1 Indledning

Nærværende rapport tjener som dokumentation af gruppens arbejde med første godkendelsesopgave. Den indeholder korrekte og afprøvede løsninger på samtlige underopgaver.

2 g1-1

På Figur 1 ses vores implementation af de grundlæggende logiske gates (AND, OR, NOT, XOR) vha. NAND-gates. Som det ses følger vores implementering metoden i lærebogens Appendix C.



Figur 1: NOT-, AND-, OR- og XOR-gates af NAND-gates

3 g1-2

Overordnet er vores 4-bit ALU (se Figur 2) implementeret som en serie af fire 1-bit ALU'er (se Figur 3) efter samme logik som den 32-bit ALU der præsenteres i Appendix C-29. Her forbindes CarryOut-outputtet fra de mindre betydende bits til CarryIn-inputtet for de mere betydende. Nedenfor er en gennemgang af de operationer 1-bit ALU'en understøtter og deres implementering:

AND, OR ALU'ens grundlæggende logiske funktioner benytter de indbyggede gates i logisim.

NOR Outputtet til NOR udregnes som NOT A AND NOT B.

Addition ALU'en benytter et Adder-modul (se Figur **??**), som vi i gruppen har implementeret vha. logisims Combinatorial Analysis-værktøj og sandhedstabellen i Figur C.53.

Subtraktion Substraktionsfunktionen benytter samme Adder, blot med én af operanderne inverteret.

Set on less than Set on less than-operationen (SLT) er en komposit operation: Først sammenlignes A og B, hvilket giver resultatet 1 hvis $A < B$, 0 ellers. Herefter sættes alle inputtets bits til 0, med undtagelse af mindst betydende bit som sættes til resultatet af sammenligningen. I forbindelse med 1-bit ALU'en giver SLT derfor ikke så megen mening.

Som det ses adskiller seriens sidste 1-bit ALU sig fra de foregående ved at understøtte yderligere funktionalitet til at undersøge for overløb (se Figur 7) samt håndtering af SLT-operationen. Denne sidste del af logikken har vi implementeret efter samme algoritme om beskrives i Appendix C-31-C-35. Bogens implementering benytter sign-bitten fra adderens output som output til SLT-operationen, hvilket undlader at tage højde for over- og underløb ved to-komplementsaritmetik. Vi håndterer dette ved at indfaktorere outputtet fra overløbsmodulet i udregningen af SLT.

Den færdige 4-bit ALU adskiller sig fra de forskellige 1-bits ALU'er ved også at give outputtet ZERO. Dette beregnes ved at føre outputtet fra samtlige operationer igennem en XNOR-gate.

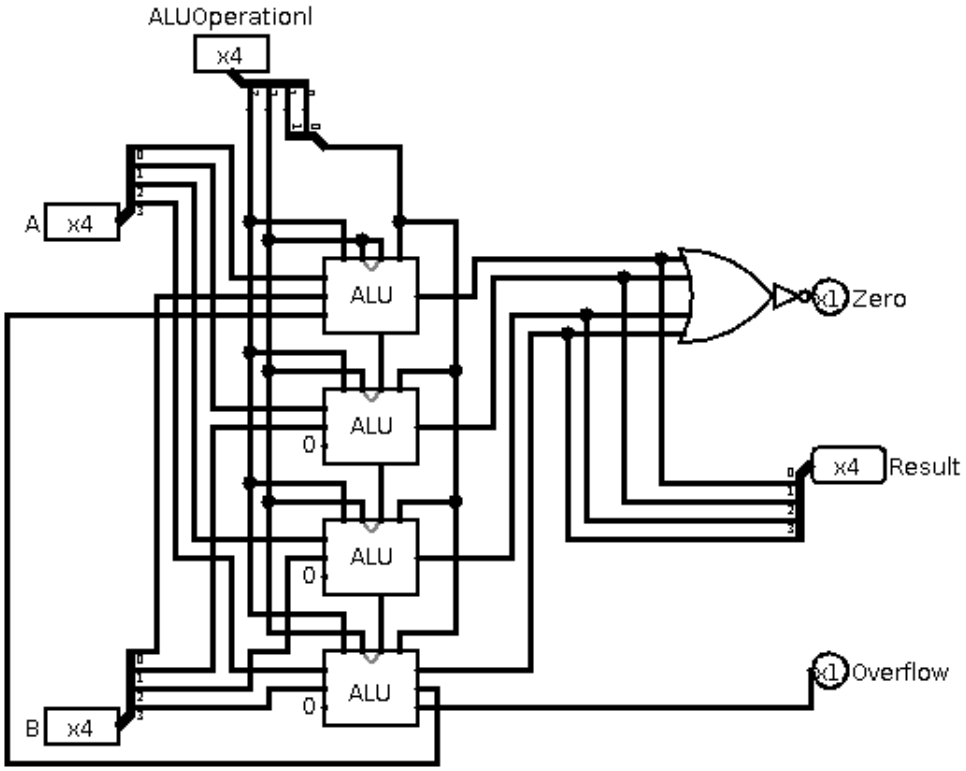
3.1 Multiplexer

3.2 Adder

3.3 Overløbskontrol

4 g1-3

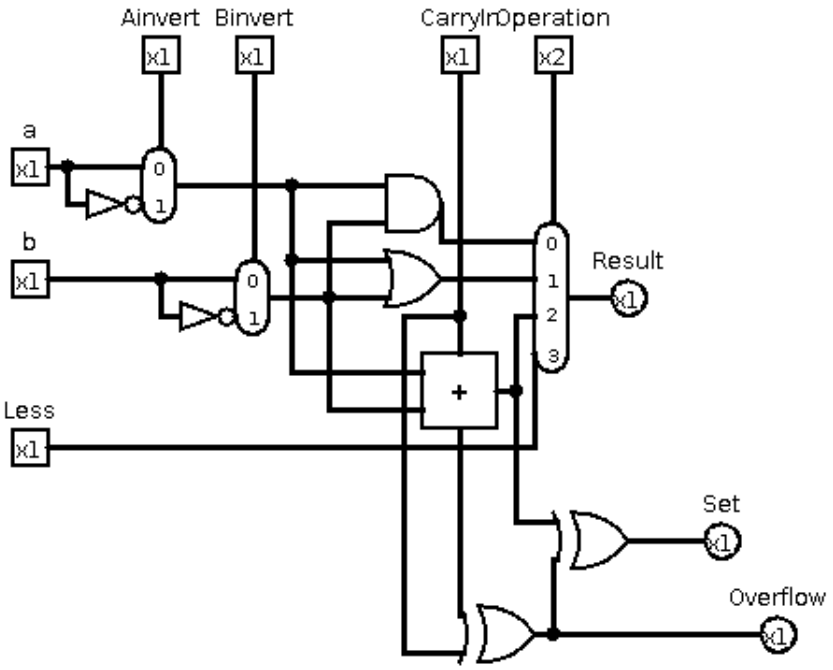
Vores 32-bit skifteenhed er implementeret som et enkelt kredsløb bestående af fem 4-IN multiplexere. Hver multiplexer svarer til en bit i shamt og vil, hvis pågældende bit i shamt er sat, skifte inputtet hhv 1, 2, 4, 8 og 16 pladser. Hvilken skifteoperationen der benyttes specificeres af



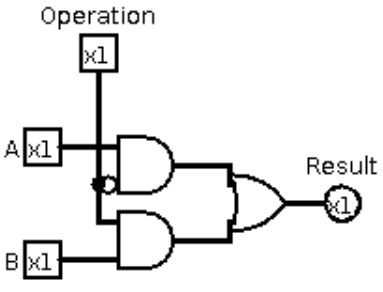
Figur 2: ALU, 4-bit

Skifteoperationerne selv er implementeret ved brug af logisims sign extender-modul således at der kopieres et antal bits (hhv. 1, 2, 4, 8 og 16) ind i inputtet. Fortegnet for det bit der kopieres ind bestemmes som selector-bitten og den

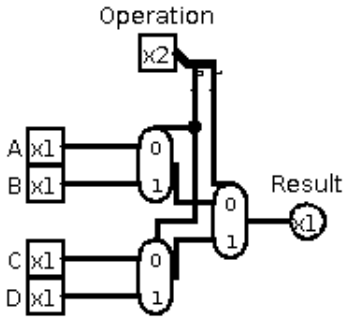
NB! Mangler argumentation for korrekthed/afprøvning!



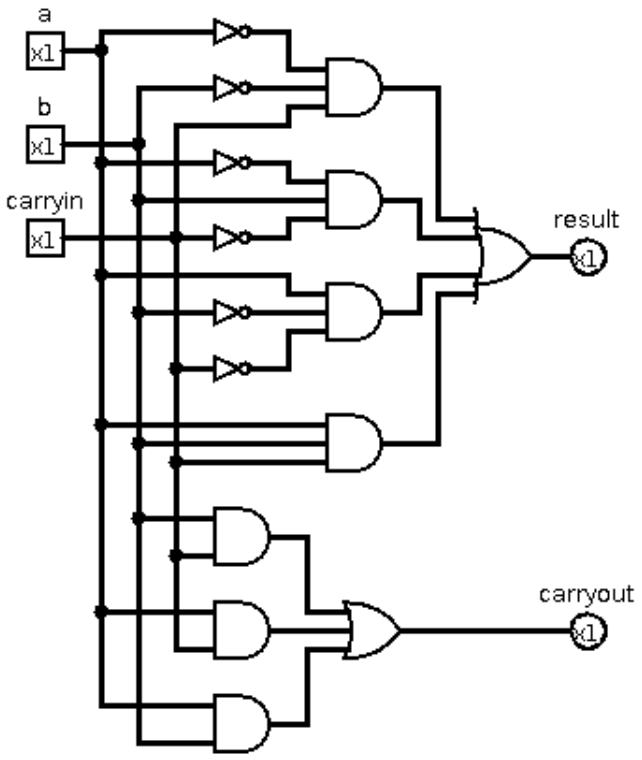
Figur 3: 1-bit ALU m. undersøgelse af overløb



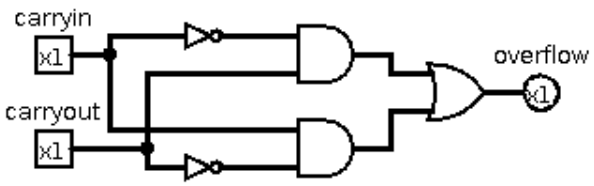
Figur 4: MUX, 2 bit IN



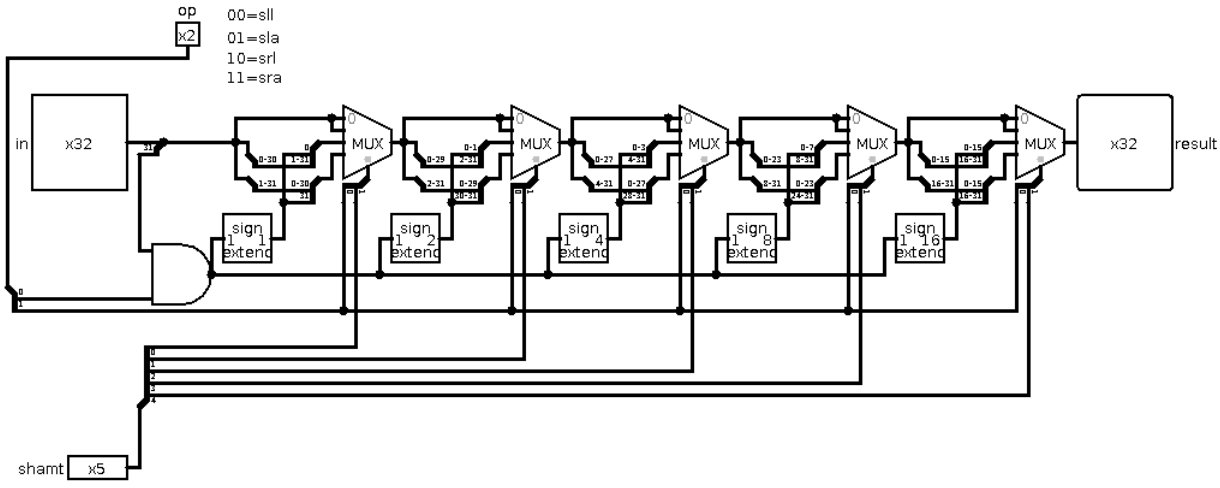
Figur 5: MUX, 4 bit IN



Figur 6: 1-bit adder



Figur 7: Logikken til undersøgelse af overløb



Figur 8: 32-bit skifteenhed