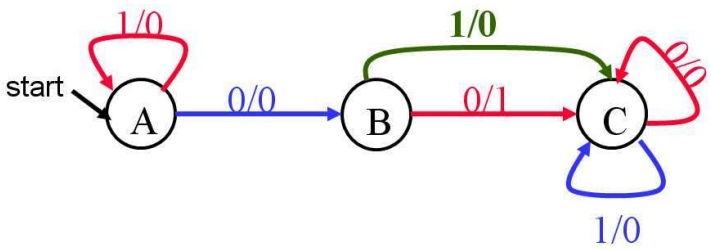
Digitale technieken Deeltoets II

André Deutz

11 januari, 2008

De opgaven kunnen uiteraard in een willekeurige volgorde gemaakt worden – geef heel duidelijk aan op welke opgave een antwoord gegegeven wordt. Onderbouw en geef uitleg voor je antwoorden . Er zijn 8 vragen en 3 bonusvragen. Elk van de 8 vragen is 12.5 punten waard. Er zijn daarnaast nog eens 3 bonusvragen. Elk van de bonusvragen is 15 punten waard. Schrijf je antwoorden op het uitgedeelde papier op met uitzondering van Vragen 5 en 8, gebruik voor het beantwoorden van deze vragen de uitgedeelde sjabloons. *Succes!*

1. Optimaliseer de Boolse functie $F(A, B, C, D) = \sum m(0, 2, 8, 9, 10, 11)$ door gebruik te maken van het Quine-McCluskey algoritme. Laat alle tussenstappen zien; geef de priem en essentiële priem implicanten aan.
2. Beschouw de volgende State Transition Diagram specificatie voor een Mealy Finite State Machine:



Figuur 1: state transition diagram; start state is A

Geef een implementatie van de FSM met behulp van rising edge D-flip-flops via de volgende stappen:

(a) Schrijf de State tabel op die hoort bij de State Transition Diagram
(b) Complementeer de State tabel met State Assignments
(c) Construeer de Waarheidstabel (Truth table) voor de Next State en Output functions

(d) Teken het logische diagram gebruikmakende van logische poorten en rising edge D-flips-flops waarmee de FSM geïmplementeerd kan worden.

3. Construeer een 4-bit ALU (arithmetic logic unit) met de volgende eigenschappen:

   - de inputs van de ALU zijn als volgt: twee 4-bit operanden, en drie control signalen (met deze laatste drie wordt de uit te voeren operatie gekozen)
   - de outputs zijn als volgt: één 4-bit resultaat, een carry_out, en een zero-output(deze output is 1 als het resultaat van de laatst uitgevoerde operatie gelijk is aan 0).
   - benodigde operaties: optellen, aftrekken, bitsgewijze AND en OR en bitsgewijze Negatie van de eerste operand.
   - Voor het optellen en aftrekken is afgesproken dat het two's complement systeem gebruikt wordt.

   Je mag aannemen dat je de beschikking hebt over een 1-bit adder. Deze heeft twee 1-bit operanden en een carry_in als input en een 1-bit som en een carry_out als output. Verder mag je gebruik maken van muxen, demuxen, encoders en decoders. Vermeld wel precies wat voor soort CLU je gebruikt (dus het vermelden van de naam van de CLU volstaat niet, je moet ook vermelden hoeveel inputs en outputs er zijn en ook hoe breed deze zijn).

   (a) Maak een tabel waarin beschreven staat welke waarden van de control input overeenkomen met de benodigde operaties.

   (b) Construeer zelf eerst een geschikte 1-bit ALU waarmee de 4-bit ALU snel gebouwd kan worden.

   (c) Construeer tenslotte de 4-bit ALU door gebruik te maken van je 1-bit ALU van de vorige vraag.

4. De volgende tabel beschrijft de instructie set van Tiny Mips – de Instruction Set Architecture (ISA) – die gebruikt moet worden voor deze vraag.

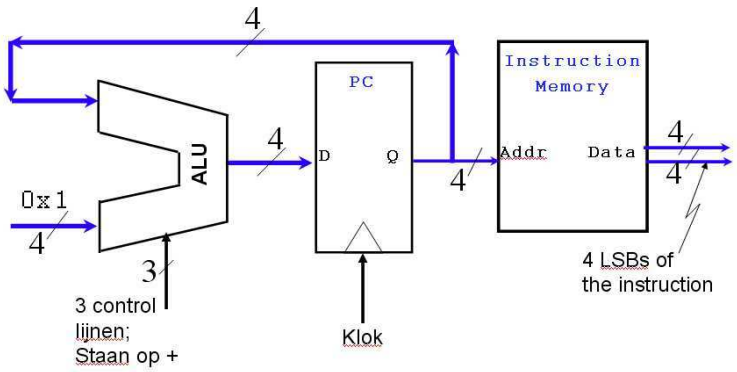
Instruction Set for Tiny Mips				
Assembly Instruction	**Opcode** (4 bits)	**Operand 1** (2 bits)	**Operand 2** (2 bits)	**Meaning**
ld r1, (r2)	0	r1	r2	r1 ← M(r2)
st r1, (r2)	1	r1	r2	M(r2) ← r1
ldi r, c	2	r	c	r ← c and two MSBs are unchanged
ldui r, c	3	r	c	r ← $4 \cdot c$ and two LSBs are unchanged
mv r1, r2	4	r1	r2	r1 ← r2
and r1, r2	5	r1	r2	r1 ← r1 AND r2
or r1, r2	6	r1	r2	r1 ← r1 OR r2
not r1, r2	7	r1	r2	r1 ← NOT r2
add r1, r2	8	r1	r2	r1 ← r1 + r2
sub r1, r2	9	r1	r2	r1 ← r1 - r2
seq r1, r2	A	r1	r2	EQ ← (r1 == r2)
breq L	B	L (4 bits)		if EQ { PC ← L }
br L	C	L (4 bits)		PC ← L
slt r1, r2	D	r1	r2	LT ← (r1 < r2)
brlt L	E	L (4 bits)		if LT { PC ← L }
halt	F	no operands		stops the eternal fetch-decode-execute cycle

Tabel 1: **The TM Instruction Set**

(a) Schrijf een TM Assembly programma dat de inhoud van datageheugen cel 7 en geheugen cel 12 verwisselt.

(b) Vertaal je assembly programma naar machine code van TM.

5. Beschouw (zie figuur) het fetch gedeelte van het datapad van de TM processor:

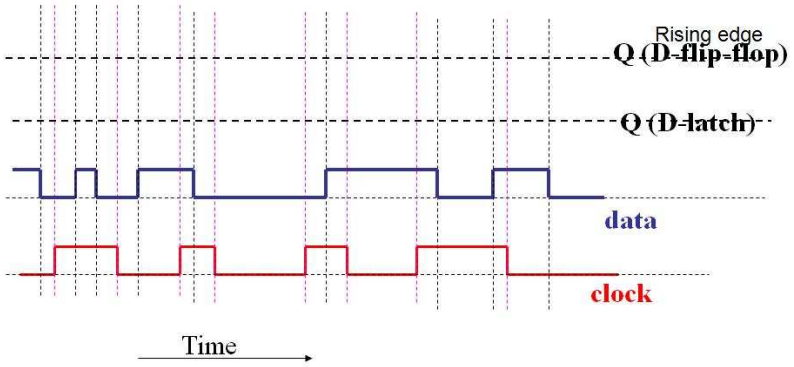


Neem aan dat je een ALU ter beschikking hebt die naast de opteloperatie minstens nog een andere operatie kent: de tweede van de twee aangeboden operanden wordt (als resultaat) naar de output gekopiëerd. Hoe kun je dan het fetch-gedeelte van het datapad aanpassen voor de non-conditional jump instructie met absolute addressering door gebruik te maken van deze ALU? Neem aan dat de 4 minst significante bits van de instructie het jumpadres voorstellen. Je mag gebruikmaken van muxen, als dat nodig is - wel vermelden wat voor soort mux je gebruikt. Vermeld ook hoe de control lijnen van de ALU ingesteld moeten worden en ook hoe de control lijnen ingesteld moeten worden van de mux(en).

6. Construeer een State Transition Diagram van het Mealy type voor een Sequence Detector. De Sequence Detector outputs telkens een 1 als elk van twee opeenvolgende inputs een 1 is (en anders wordt er een 0 geoutput). Bijvoorbeeld: als de input sequence 0111101... is, dan is de output sequence 0011100...

7. Construeer het Logische diagram voor een geheugen met 3 woorden. De bitbreedte van elk van de woorden is 2. Als invoer hebben we natuurlijk twee datalijnen, het adres (ook twee lijnen). Daarnaast ook een Write-enable, een Read-enable, eventueel een chip-select. Voor het 1-bit geheugen gebruiken we een rising-edge D-flip-flop. De schrijfdata en de uit te lezen data maken gebruik van dezelfde bus (die 2-bit breed is).

8. Beschouw het data signaal en het klok signaal in het volgende diagram. Teken het bijbehorende Q-signaal voor zowel een d-latch en als een rising edge d-flip-flop.



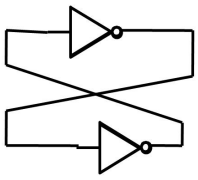
Figuur 2: data en clock signaal diagram

9. **Bonusvraag 1:**

   (a) Wat zijn de drie gebruikelijke manieren om geklokte flip-flops te klokken (te triggeren)?

   (b) Beschrijf de redenen waarom men verschillende manieren van klokken van flip-flops nodig heeft.

10. **Bonusvraag 2:**
    Beschrijf van het circuit in Figuur 3 de stabiele en metastabiele toestanden.



Figuur 3: gekoppelde NOT gates

11. **Bonusvraag 3: het Algoritme van Booth**

(a) Op welk idee berust het Algoritme van Booth?

(b) Beschrijf het Algoritme van Booth.

(c) Bewijs dat het Algoritme van Booth ook goed werkt wanneer men vermenigvuldiging van zowel positieve en negatieve getallen (gerepresenteerd in het two's complement systeem) op het oog heeft.