You are welcome to visit me from 12:30 to 13:30 to check your exam score on Monday, 31st October 2016

Solutions for the exam of TSEA26 at 2016-10-28

Question 1: MAC (15p)

Draw a schematic and a control table for a MAC unit with the following operations:

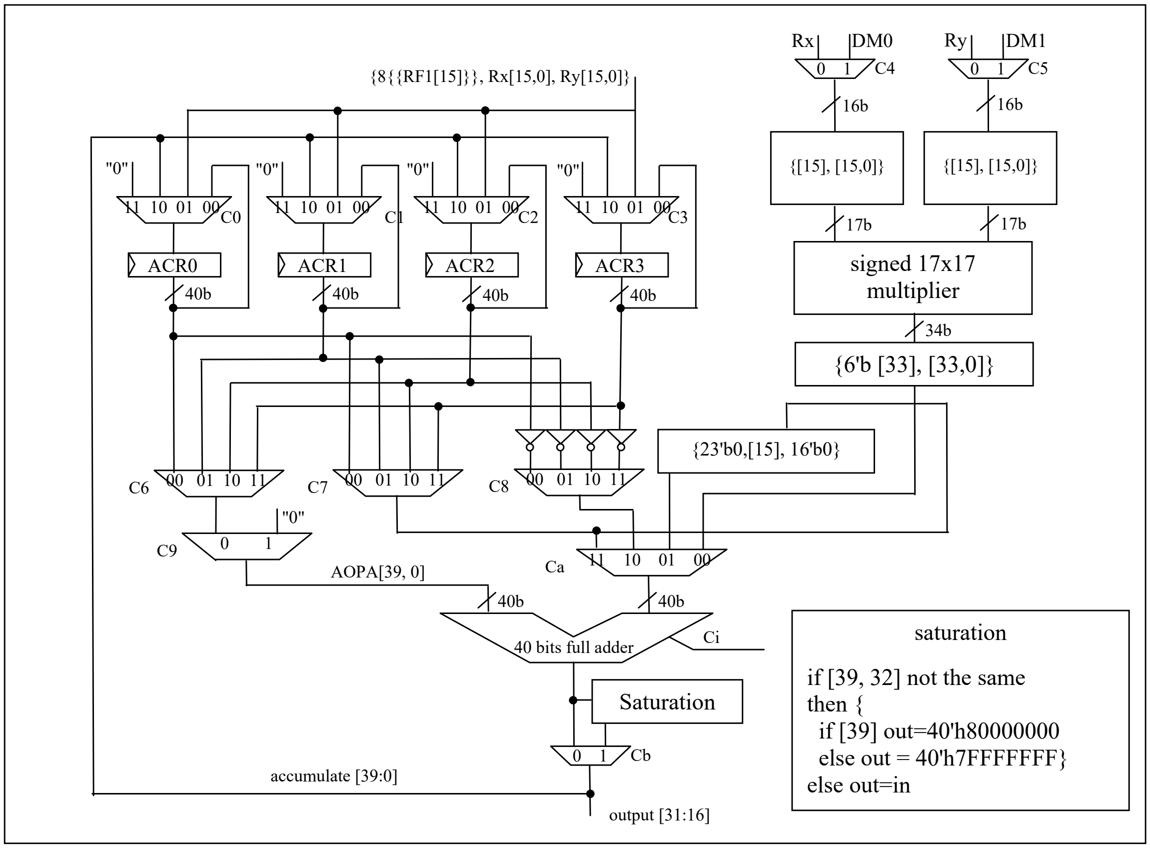
- `OP0: NOP`
- `OP1: ACRx = 0`
- `OP2: ACRx = {{8{RF1[15]}}, RF1[15:0], RF2[15:0]}`
- `OP3: ACRx = RF1[15:0] * RF2[15:0] (signed integer multiplication)`
- `OP4: ACRx = ACRx + DM0[ap0] * DM1[ap1] (signed integer MUL)`
- `OP5: ACRx = ACRy + ACRz`
- `OP6: ACRx = ACRy - ACRz`
- `OP7: RF1[15:0] = SAT(ROUND(ACRy)) (move ACRy[31:16] to RF1)`

Constraints, inputs, outputs, and proposals:

- Your MAC unit should have 4 accumulator registers (0, 1, 2, and 3). Each accumulator register is 40 bits wide
- RF is 16 bit input/output from/to the general register file
- You are proposed to use 17b x 17b signed multiplier
- You need to offer bit accurate annotations on connections
- x, y, z: 2 bit wide inputs from the instruction decoder that selects the appropriate accumulator register
- To simplify your control table, you only need to specify when x=0, y=1, and z=2.
- And of course, whatever clock signals and control signals you deem necessary.

Solution 1:

Q1: Draw a schematic and a control table for the MAC unit:



Control table (suppose that ACRx is ACR0; ACRy is ACR1; and ACRz is ACR2)

OP	code	Operation	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Ca	Cb	Ci
0	000	NOP	00	00	00	00	x	x	x	x	x	x	x	x	x
1	001	0	11	00	00	00	x	x	x	x	x	x	x	x	x
2	010	{8h R1[15], R1[15,0], R2[15,0]}	01	00	00	00	x	x	x	x	x	x	x	x	x
3	011	Signed Integer MUL	10	00	00	00	0	0	x	x	x	1	00	0	0
4	100	Signed Integer MAC	10	00	00	00	1	1	00	x	x	0	00	0	0
5	101	ACRy+ACRz	10	00	00	00	x	x	01	10	x	0	11	0	0
6	110	ACRy-ACRz	10	00	00	00	x	x	01	x	10	0	10	0	1
7	111	R1=S(R(ACRy))	00	00	00	00	x	x	01	01	x	0	01	1	0

You can as well finish the rest of the table by selecting any ACR0-3 to be the sources and the result

Question 2: ALU (5p)

Please design a 16b in/out arithmetic computing unit using only one adder and simple logic component such as multiplexer and logic gates. Please design the circuit in detail either using HDL code or detail schematic drawing with complete connections and width annotations on each connection. The instruction subset of the arithmetic unit is given in the following table. The adder uses operands A and B from the general register file. The width of the processor memory bus is 16 bits. Two operands can be supplied to the arithmetic unit the same clock cycle. Specify all control signals and finish a binary control table.

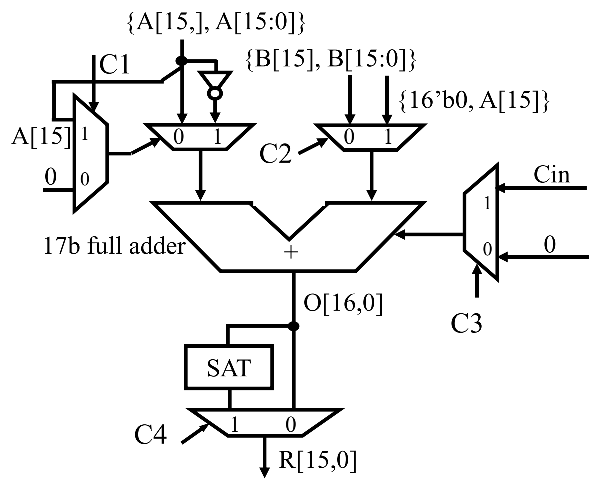
	Instructions	*Function*	*OP*
1	ADD SAT	A + B with saturation	*000*
2	ADD	A + B without saturation	*001*
3	ADD CIN SAT	A + B + Cin with saturation	*010*
4	ADD CIN	A + B + Cin without saturation	*011*
5	*ABS(A)*	*ABS(A) Absolute operation*	*100*

Solution 2:

schematic and a control table for the ALU unit:

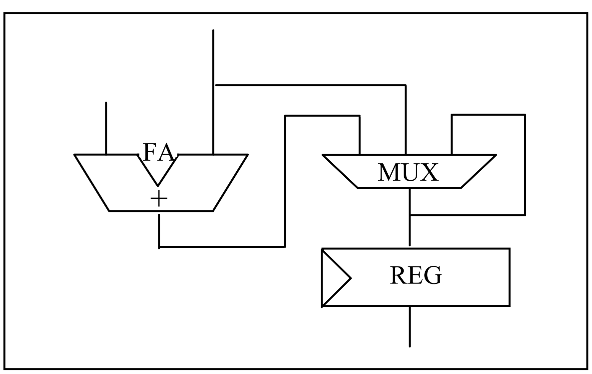
OP	Opcode	C1	C2	C3	C4
A+B	000	0	0	0	0
S(A+B)	001	0	0	0	1
A+B+C	010	0	0	1	0
S(A+B+C)	011	0	0	1	1
ABS	100	1	1	0	1

```
SAT function
If O[16] = O[15] then R=O[15:0]
Elseif O[16] R=16'h8000
Else R=16'h7FFF
```



Question 3: General questions (12p)

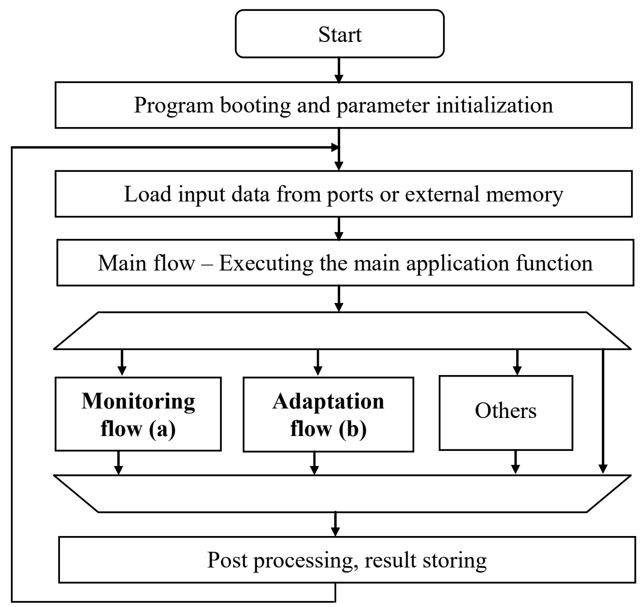
3.1. Find the not tolerable error in the following figure (1p)



3.2. Using (Gajski) Y-chart and three sentences to describe an Application Specific Instruction-set Processor (1p)

3.3. Select 3 data samples (suppose the three samples are already in R1, R2, and R3) to verify the "16b absolute instruction", ABS (Rx), x=1, 2, 3,..... . (3p).

3.4. Please describe (3p)
3.4.1: What is the monitoring flow marked (a) in the following flowchart? (1p)
3.4.2: What is the adaptation flow marked (b) in the following flowchart? (1p)
3.4.3: Please find a DSP processor which does not require step (a) and (b) because of the special hardware and data type (1p).



Flowchart of Q3.4

3.5. (3p) If multiply is executed in one clock cycle, and accumulation is executed in another clock cycle, which of the following program(s) based on pseudo assembly codes (from 1 to 4) is/are correct (1p)? Describe the errors if they are (it is) not correct (2p).

3.5.1: Program 1:
```
{1:  ACR1 <= ACR1 + R1*R2;
 2:  R3   <= round(saturation(ACR1)); }
```

3.5.2: Program 2:
```
{1:  ACR1 <= ACR1 + R1*R2;
 2:  R3   <= saturation(round(ACR1)); }
```

3.5.3: Program 3:
```
{1:  ACR1 <= ACR1 + R1*R2;
 2:  NOP;
 3:  R3   <= round(saturation(ACR1)); }
```

3.5.4: Program 4:
```
{1:  ACR1 <= ACR1 + R1*R2;
 2:  NOP;
 3:  R3   <= saturation(round(ACR1)); }
```

3.6 Based on Senior DSP processor used in the lab work, when we accelerate for motion estimation, why we cannot run multiple points for SAD in a clock cycle? (1p).

Solution 3:

Q3: general questions
3.1.
MUX in and out is a combinational loop
3.2.
Behavior: A processor platform to accelerate custom functions and the platform keeps its scoped flexibility.
Structure: Custom architecture accelerations in datapath, data access path, and control path
Physical: To reach custom performance, power consumption, and silicon cost
3.3.
ABS R2, R1 // R1=any positive value, the result shall be the R1=R2
ABS R2, R1 // R1=any negative value instead of 16'h8000, the result shall be R2 = inv(R1) + 1
ABS R2, R1 // R1=16'h8000, the result shall be 16'h7FFF
3.4.
To measure the values at the points for scaling
Scaling according to the measurements
Floating point DSP processor
3.5.
1. wrong order for rounding and saturation, as well data hazard
2. data dependency hazard
3. wrong order saturation and rounding
4. correct
3.6.
We do not have sufficient memory bandwidth

Question 4: Address Generator Unit (10p)

Draw a schematic and a control table for an AGU, including essential peripheral circuits for M0, and M1, supporting two address pointers for both DM0(ap0≤65535) and DM1(ap1≤65535), and supporting for the following addressing modes:

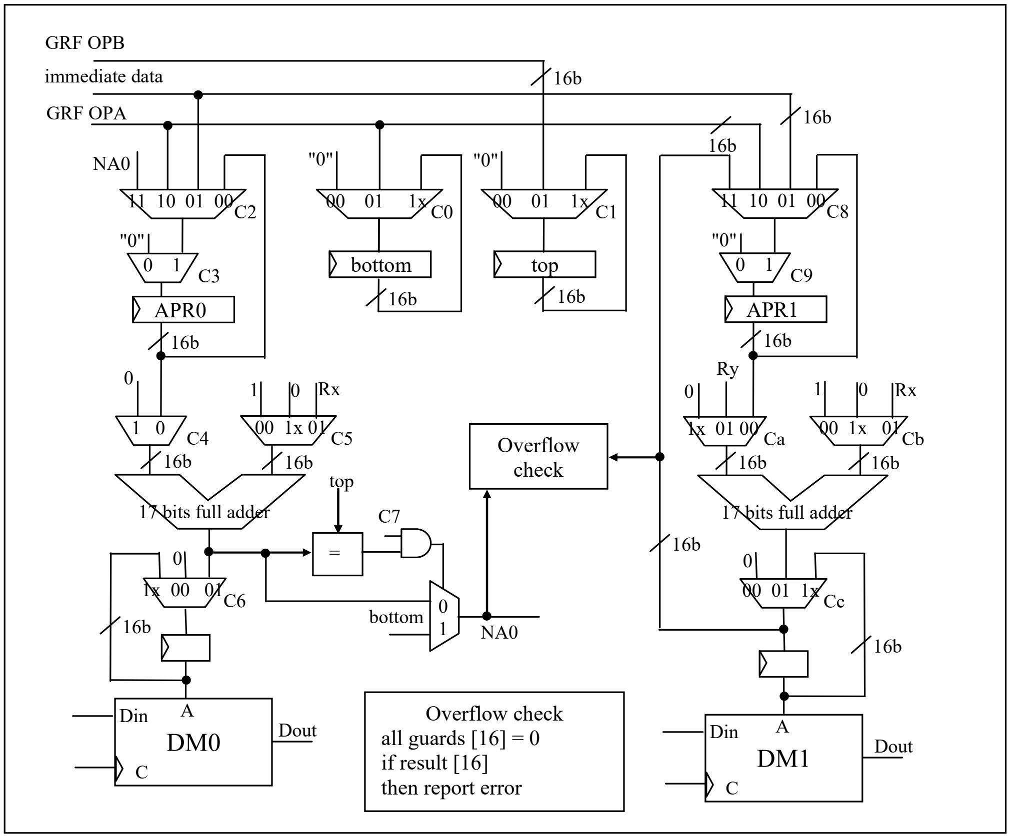
	op	*code*	*description*
0	Reset	0000	Clear registers
1	NOP	0001	No operation
2	Load Apr0	0010	From general register (Rx, Ry)
3	Load Apr1	0011	From general register (Rx, Ry)
4	Set Apr0	0100	Using instruction carried immediate data
5	Set Apr1	0101	Using instruction carried immediate data
6	Load Rtop and bottom	0110	From general register Top=Rx, Bottom=Ry
7	M1 [Rx+Ry]	0111	Look up table index addressing Y=LUT[x]
8	M0 [Rx]	1000	M0 addressed by a general register
9	M1 [Rx]	1001	M1 addressed by a general register
10	M0 [Apr0]	1010	M0 addressed Apr0
11	M1 [Apr1]	1011	M1 addressed Apr1
12	M0 [Apr0++]	1100	Post incremental addressing for M0

13	M1 [Apr1++]	1101	Post incremental addressing for M1
14	M0 [Apr0++], M1[Apr1++]	1110	Supporting vector multiplication
15	M0 [Apr0%++], M1[Apr1++]	1111	Supporting convolution

Inputs are from 16b general registers (operand A = Rx and operand B = Ry), and 16b immediate data. Outputs are addresses to memories DM0 and DM1.

Solution 4:

Schematic



Control table

	op	code	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Ca	Cb	Cc
0	Reset	0000	00	00	00	0	x	x	00	0	00	0	x	x	00
1	NOP	0001	1x	1x	00	1	x	x	1x	0	00	1	x	x	1x
2	Load Apr0	0010	1x	1x	10	1	x	x	1x	0	00	1	x	x	1x
3	Load Apr1	0011	1x	1x	00	1	x	x	1x	0	10	1	x	x	1x
4	Set Apr0	0100	1x	1x	01	1	x	x	1x	0	00	1	x	x	1x
5	Set Apr1	0101	1x	1x	00	1	x	x	1x	0	01	1	x	x	1x
6	Load Rtop and bottom	0110	01	01	00	1	x	x	1x	0	00	1	x	x	1x
7	M1 [Rx+Ry]	0111	1x	1x	00	1	x	x	1x	0	00	1	01	01	01
8	M0 [Rx]	1000	1x	1x	00	1	1	01	01	0	00	1	x	x	1x
9	M1 [Rx]	1001	1x	1x	00	1	x	x	1x	0	00	1	1x	01	01
10	M0 [Apr0]	1010	1x	1x	00	1	0	1x	01	0	00	1	x	x	1x
11	M1 [Apr1]	1011	1x	1x	00	1	x	x	1x	0	00	1	00	1x	01
12	M0 [Apr0++]	1100	1x	1x	11	1	0	00	01	0	00	1	x	x	1x
13	M1 [Apr1++]	1101	1x	1x	00	1	x	x	1x	0	11	1	00	00	01
14	M0 [Apr0++], M1[Apr1++]	1110	1x	1x	11	1	0	00	01	0	11	1	00	00	01
15	M0 [Apr0%++], M1[Apr1++]	1111	1x	1x	11	1	0	00	01	1	11	1	00	00	01

Question 5: PFC (8p)

Design part of the control path: The design shall include functions:

1. PC[15:0] <= 0; Reset, and starts executing at address 0x0000 after reset
2. PC[15:0] <= PC+1; default
3. PC[15:0] <= immediate [15:0]; unconditional jump, 16b target address is carried by the jump instruction.

Outputs of Pipeline registers are:

Pipeline	*Register Output Description*
PC_out	Program memory address
I_buffer_out	Fetched instruction (to be decoded)
Decoded_control_signals	Registered control signals
Operands	Available operands
ALU_results	Available results (flags are available here)

5.1. (4p) Draw a simplified pipeline schematic with necessary annotations. When an unconditional jump is decoded, the hardware shall automatically insert NOP(s). You shall add only necessary and minimum number of NOP(s).

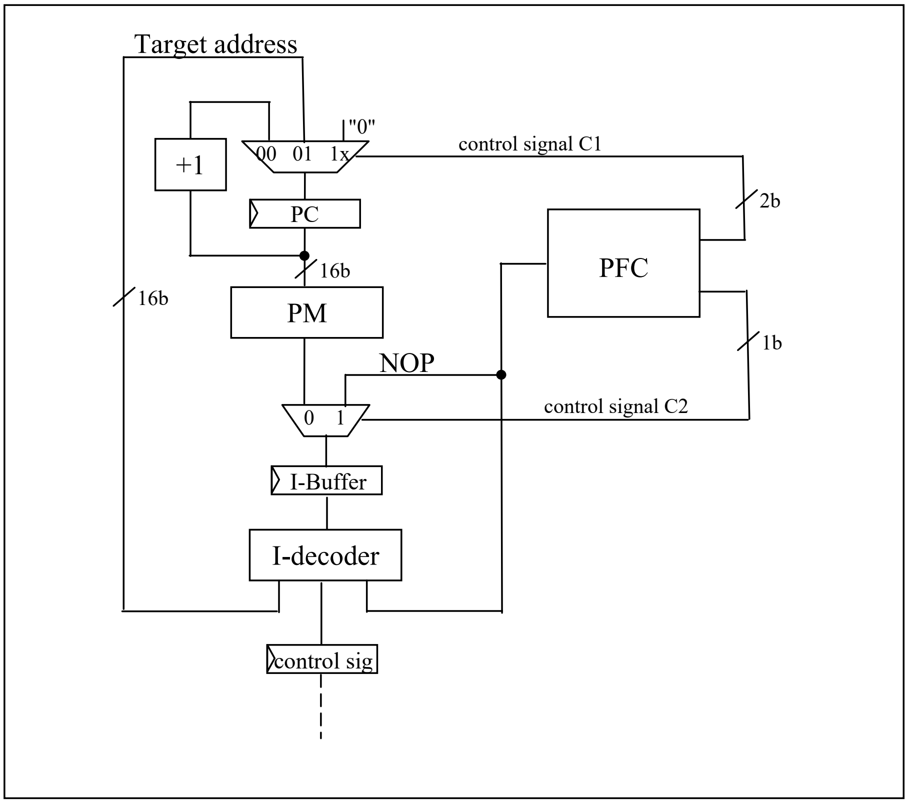
5.2. (2p) Design a pipeline table for following unconditional jump pseudo codes

```
            JUMP IMMEDIATE
            XXX
            YYY
            ZZZ

            ...

IMMEDIATE:  ADD
```

5.3. (2p) Write pseudo codes for necessary control signals you specified in your schematic.

Solution 5:

Schematic



Pipeline table

Clock	PC_out	I_Buffer_out	Decoded_Control_signals	Operands	ALU_Results
1	Jump				
2	NOP	Jump			
3	ADD	NOP	Jmp		
4		ADD	NOP	Jump	
5			ADD	NOP	Jump
6				ADD	NOP
7					ADD

Pseudo codes for C1 and C2 in PFC

```
If reset
    Then C1 = 1x
Elseif unconditional jump
    Then C1 = 01
Else C1 = 00

If unconditional jump
    Then C2 = 1
Else C2 = 0
```