Design of Embedded DSP Processors, TSEA26

Date	2011-08-23
Room	
Time	14:00-18:00
Course code	TSEA26
Exam code	TEN 1
Course name	Design of Embedded DSP Processors
Department	ISY, Department of EE
Number of questions	5
Number of pages (including this page)	6
Course responsible	Andreas Ehliar
Teacher visiting the exam room *Phone number during the exam time*	Olle Seger 013 - 28 2159
Visiting the exam room	Around 15 and 17
Course administrator	Ylva Jernling, 013-282648, ylva@isy.liu.se
Permitted equipment	None, besides an English dictionary
Grading	**Points** **Swedish grade** 41-50 5 31-40 4 21-30 3 0-20 U

- Exams are normally corrected within 10 working days. However, due to parent leave it is likely that it will take longer than normal to correct the exam this time. *For more information about the exam correction, please see the course homepage.*
- You can answer in English or Swedish. Don't write answers on the exam sheet.
- **When designing a hardware unit you should attempt to minimize the amount of hardware used.** (Unless otherwise noted in the question.)
- The width of data buses and registers must be specified unless otherwise noted. Likewise, the alignment must be specified in all concatenations of signals or buses. When using a box such as *"SATURATE"* or *"ROUND"* in your schematic, you must (unless otherwise noted) describe the content of this box! (E.g. with RTL code). You can assume that all numbers are in two's complement representation unless otherwise noted in the question.
- In questions where you are supposed to write an assembler program based on pseudo code you are allowed to optimize the assembler program in various ways as long as the output of the assembler program is identical to the output from the pseudo code. You can also (unless otherwise noted in the question) assume that hazards will not occur due to parts of the processor that you are not designing.

Question 1: (5p)

Draw a schematic and a control table for a program flow control unit that supports the operations listed in the table below. The allowed inputs and outputs are also listed in a table below. At reset the system should start executing at address `0x800`. You don't have to worry about pipelining issues such as delay slots when solving this question. The program counter is 14 bits wide.

Name	Operation
OP1	Branch on equal
OP2	Branch on not equal
OP3	Branch always

Inputs and outputs

Direction	Name	Comment
Input	IMM[13:0]	The immediate data field from the instruction word
Input	Z	Zero flag from the ALU
Input	RST	Reset signal
Input	*Your choice*	Control signals created by you in your control table
Output	TO_PM[13:0]	The address which is sent to the program memory

Question 2: Arithmetic Unit (15p)

a) Draw a schematic and a control table for an arithmetic unit with the following operations: (6p)

Name	Operation
OP1	RESULT = OpA + OpB
OP2	RESULT = OpA - OpB
OP3	RESULT = SAT(OpA - OpB)
OP4	RESULT = SAT(ABS(OpA))
OP5	RESULT = SAT(ABS(A - B))
OP6	RESULT = BITREVERSE(OpA)

Inputs and outputs

Direction	Name	Comment
Input	OpA[15:0]	Input from the register file
Input	OpB[15:0]	Input from the register file
Input	*Your choice*	Control signals created in your control table
Output	RESULT[15:0]	Output to the write back mux

Hint: Read through part b on the next page before you start to draw a schematic as you will need to create a more complicated schematic if you choose to answer this part.

b) The function `saturate_values()` needs to be supported on your DSP processor and your task is to add the ALU functionality necessary to support it.

You need to figure out which instructions you will need in the ALU to support the function. You need to add support for those instructions to the **schematic** and you also need to update the **control table**. If you need to connect the ALU to other components in the DSP processor (such as DM0) your schematic must clearly show how you do this! Finally you need to translate `saturate_values()` into **assembler**.

You can assume that the other parts of the processor have all features necessary to support this function. (E.g., the AGU has appropriate addressing modes, the PFC has a repeat instruction, etc.)

```
; r0 contains ptr
; r1 contains maxval
; r2 contains minval
function saturate_values(ptr, maxval, minval)
    repeat 160
        ; x is a 16 bit temporary value
        x = DM0[ptr]

        if x > maxval then
            DM0[ptr] = maxval
        else if x < minval then
            DM0[ptr] = minval
        end if
        ptr = ptr + 1
    endrepeat
endfunction
```

There are two sets of constraints for this part of the question. It is up to you which alternative you want to design for, but you will only be able to get the maximum points of this question if you choose to fulfill the constraints listed under alternative 2 below.

Constraint alternative 1 (worth 5 points):

- The function `saturate_values` need to execute in less than 380 clock cycles.
- DM0 is a single port memory which is 16 bit wide.

Constraint alternative 2 (worth 9 points):

- The function `saturate_values` need to execute in less than 240 clock cycles.
- You may change the width of DM0 to 32 bit if you would like to. (You may not change the number of ports however, it is still a single ported memory.)
- The parameter ptr which is passed to the function is guaranteed to be even.

Question 3: MAC unit (15p)

You should design a MAC unit which is optimized to run the `do_filter()` and `do_small_fir()` functions. The code and clock cycle constraints for these functions are shown below and on the next page.

Allowed inputs and outputs

Direction	Name	Comment
Input	DM0_result[15:0]	Data that have been read from DM0
Input	DM1_result[15:0]	Data that have been read from DM1
Input	OpA[15:0]	An operand from the register file
Input	OpB[15:0]	The other operand from the register file
Input	*Your choice*	Control signals created in the control table
Input	clk	The system clock
Output	TO_RF[15:0]	This is sent to the register file writeback port
Output	TO_DM0[15:0]	This is sent to the write port of DM0
Output	TO_DM1[15:0]	This is sent to the write port of DM1

```
; ptr1 is passed in register r0
; ptr2 is passed in register r1
; ptr3 is passed in register r2
; flag is passed in register r3
; Returns data in r0 and potentially in DM0[ptr3]
function do_filter(ptr1, ptr2, ptr3, flag)
    tmp = 0
    repeat 100
        ; Note: It is not a mistake that DM0 is used twice in the
        ; fractional multiplications below.
        if flag == 0 then
            tmp = tmp + DM0[ptr1] * DM0[ptr2]
        else
            tmp = tmp + ABS(DM0[ptr1] * DM0[ptr2])
        endif

        ptr1 = ptr1 + 1
        ptr2 = ptr2 + 1
    endrepeat

    if ABS(tmp) >= 0x80000000 then
        tmp = tmp >> 4      ; Arithmetic right shift
        DM0[ptr3] = 1
    end if
    return SAT(tmp)  ; Return value in register r0
endfunction
```

Constraint: This function should execute in a maximum of 260 clock cycles.

```
; ptr1 is passed in register r0
; ptr2 is passed in register r1
; tap1 is passed in register r2
; tap2 is passed in register r3
; tap3 is passed in register r4
; Returns data in DM0 in an array pointed to by ptr2
function do_small_fir(ptr1, ptr2, tap1, tap2, tap3)
    tmp = 0

    oldval1 = DM0[ptr1]
    oldval2 = DM0[ptr1+1]
    ptr1 = ptr1 + 2
    repeat 99
        ; tap1-tap3 are 16 bit fractional values.
        ; val, oldval1, and oldval2 are 16 bit fractional values

        val = DM0[ptr1]
        ptr1 = ptr1 + 1

        tmp = oldval1 * tap1 + oldval2 * tap2 + val * tap3

        oldval1 = oldval2
        oldval2 = val

        ; We should store a fractional value to DM1 here
        DM1[ptr2] = SAT(tmp)
        ptr2 = ptr2 + 1
    endrepeat
endfunction
```

Constraint: This function should execute in at most 440 clock cycles

a) Select a suitable instruction set for your MAC unit. You can assume that the other units in the DSP processor has all features required to execute these programs under the listed constraints. For example, the AGU:s have a post increment addressing mode, the PFC has a repeat instruction, and so on. (If you answer part b, c, *and* d you don't have to write a specific answer to part a since it is obvious which instructions you have selected anyway). (3p)

b) Translate `do_filter` and `do_small_fir` into assembly code that fulfills the constraints listed above. (5p)

c) Draw a schematic for your MAC unit. Your accumulator(s) should be 40 bit wide with 8 guard bits. (6p)

d) Draw a control table for your MAC unit. (2p)

Question 4: Address Generator Unit (10p)

You are designing an address generator unit (AGU) for a certain DSP processor. The application engineers desire the following addressing modes:

	Address sent to memory	Operation on address register
OP1	AR[15:0]	NOP
OP2	AR[15:0]+RF[15:0]	NOP
OP3	AR[15:0]+1	AR[15:0] = AR[15:0] + 1
OP4	AR[15:0]	AR[15:0] = AR[15:0] + 1
OP5	AR[15:0]-1	AR[15:0] = AR[15:0] - 1
OP6	AR[15:0]	AR[15:0] = AR[15:0] - 1
OP7	AR[15:0]+SIGNEXTEND(IMM[7:0])	NOP
OP8	RF[15:0]+SIGNEXTEND(IMM[7:0])	NOP

In addition, the application engineers also need to access circular buffers or FIFOs in an efficient manner. Therefore they require support for some sort of modulo addressing mode with a step size of one. You need to figure out what operations to add to the AGU in order to support this.

a) Draw a schematic for an AGU supporting all operations outlined above plus modulo addressing. When drawing your schematic you need to show how you connect your AGU to the memory subsystem. (8p)

b) Draw a control table for your AGU including all operations listed above plus all operations required for modulo addressing. (2p)

Question 5 (5p)

a) A three tap FIR filter has the coefficients of 0.25, 0.125, and -0.25. The inputs and outputs of this FIR filter is supposed to be in fractional format. How many guard bits are required in the accumulator in the MAC unit? (1p)

b) Pipelining is an important technique that is used to great effect in both DSP processors and regular processors. However, this optimization does not come without costs. Give two reasons why it is not a good idea to increase the number of pipeline stages beyond a certain point. (E.g., 10 pipeline stages might be a good idea whereas 30 pipeline stages might be a bad idea in terms of performance.) Use at most 6 sentences. (2p)

c) Create a unit which will round a 16[1] bit two's complement number in Q5.15 format to Q5.10 format. You should use the rounding algorithm which has been discussed throughout the course. (The rounding algorithm discussed in the course is called round to nearest. In case the number in Q5.15 format is equally close to two Q5.10 numbers you should round up towards positive infinity.) (2p)

[1]Errata: This was a typo in the exam, it should really be a 20 bit number!