

**ECE 574: Modeling and synthesis of digital systems using Verilog and VHDL  
Fall Semester 2009**

**Design of a DSP processor  
Report and Signoff due Week 11 (November 19<sup>th</sup>)**

***Important Note:***

You have 4 weeks to complete this project but remember the final project is also due by the end of the class and so you will need to work on both projects in parallel.

***Description***

Your goal is to design a simple DSP processor. It will have four 8-bit registers (R0, R1, R2, R3) and can accept an 8-bit instruction. As well as the usual arithmetic, logical, and multiply and accumulate (MAC) instructions, it can also carry out load and store operations using the external SRAM on the Nexys2 board.

You should set the next instruction to be entered on the DIP switches and then press push button BTN0 to enter the instruction and execute it.

Configure the external SRAM memory (using the address and control lines) to provide 256 8-bit locations.

The following eight operations should be supported:

LD	R,#	Load register with immediate data
LDM	Rs, (Rd)	Load register with data at memory location pointed to by register
STM	Rs, (Rd)	Store register to memory location pointed to by register
AND	Rd,Rs	'AND' operation of two registers
OR	Rd,Rs	'OR' operation of two registers
ADD	Rd,Rs	addition operation of two registers
MUL	Rd,Rs	multiply operation of two registers
MAC	Rd,Rs	multiply and accumulate two registers

In each case the instruction should specify the two registers to be used, or a register and immediate data. Any combination of registers for source and destination should be supported. For example

```
LD    R2, 07    ; load R2 with immediate data value 7
LD    R3, 05    ; load R3 with immediate data value 5
ADD   R2, R3    ; R2 = R2 + R3
MAC   R0, R1    ; R0 = R0 + (R0*R1)
```

The load/store memory instructions should specify a source or destination register and a register to be used as the address of the memory (register indirect addressing).

```

LDM R0,(R3)      ; load R0 with memory location pointed to by R3
STM R2,(R1)      ; store R2 to memory location pointed to by R1

```

Develop your own instruction coding – use 3-bits for the opcode, 2 bits for the source and 2-bits for the destination register field. (This allows 3 bits for immediate data for load immediate instructions.)

Use the seven segment displays to show the contents of various registers and the memory. When no buttons are pressed show the contents of R1 and R0 registers. When BTN1 is pressed show the contents of R2 and R3 registers. When BTN2 is pressed show the contents of the SRAM location with the address set by the DIP switches,

Your VHDL design must include a package, and use at least one subprogram (function).

Write a test bench to show the correct operation of your DSP design using the following instructions (values are in hex). Use *assert* statements to verify the outputs are correct – but also deliberately force an assertion failure to show the simulation output.

```

LD    R0,1        ; load R0 with 1
LD    R1,2        ; load R1 with 2
LD    R2,3        ; load R2 with 3
ADD   R1,R2       ; R1 = R1 + R2 = 2 + 3 = 5
MUL   R2,R1       ; R2 = R2 * R1 = 3 * 5 = F
MAC   R1,R2       ; R1 = 5 + (5 * F) = 50
STM   R2,(R0)     ; store R2 (F) into [R0] (SRAM address 1)
STM   R0,(R1)     ; store R0 (1) into [R1] (SRAM address 50)
LDM   R3,(R0)     ; R3 = F
LDM   R2,(R1)     ; R2 = 1
ADD   R2,R3       ; R2 = R2 + R3 = 10

```

You will need to create a simple model of the SRAM memory to work with your test bench (see example provided in class)

Also use the Nexys board to demonstrate (by week 11) the correct operation of the same piece of code. You will need the usual signoff sheet and VHDL listing. The TA may check that other operations also work correctly.

Include the same level of documentation as required for your VGA and Serial design. See the “preparing the written report” guidelines. You should include an introduction giving an overview of your complete design. This should include such information as a block diagram, explanations of the main modules and features of your design and any other descriptions to help my (or anyone else’s) understanding of how your design works.