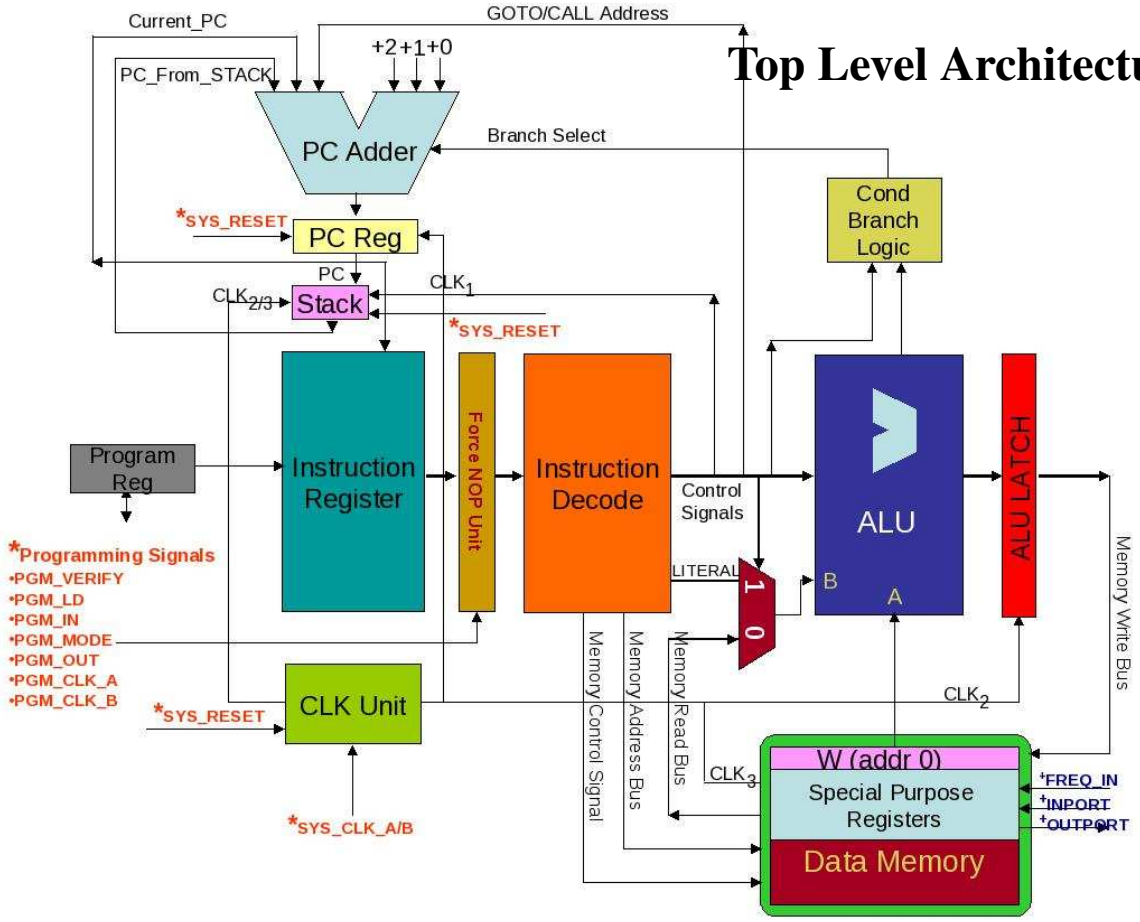
LAB Assignment #7 for ECE 443

Assigned: Wed., Oct 21, 2009
Due: Mon., Oct 28, 2009

Description: Build a (temporary) Instruction Register and Instruction Decoder of the microcontroller

We need to start talking about the big picture in this lab so you know where we are going. The microcontroller we are designing was originally designed by Dhruva Acharyya. The top-level architecture he derived for the microcontroller is shown below.



Red: external signals to control programming the microcontroller
Blue: external signals for runtime communication/operation

Features:

- Separate instruction and data memory
- Only 19 instructions
- 6 clk cycles / instruction cycle
- 8 special purpose registers
- 24 General purpose registers
- 1 output port (12-bits)
- 1 input port (12-bits)
- 1 counter (implemented as 2 registers)

- All registers are memory mapped
- Hardware stack (6 deep)
- Instructions and datapath are 12 bit wide
- Hybrid of pre-programmed ROM and read/write memory

Instruction Set

Instruction Set

INSTR	OPCODE	FUNC/LITERAL/F	DESCRIPTION
ALUWF	000	O O O D F F F F F F	ALU operation with W with F and store in F or W (determined by D). Operation determined by OOO value. (mem: read = 1 ,write = 1)
MOVF	001	X X X D F F F F F F	MOVE data from F to W or W to F (determined by D) (mem: read = 1 ,write = 1),D=1: F->W, D=0: W->F
CALL	010	A A A A A A A A A A	Unconditional Branch to a particular PC. PC is stored in the stack (mem: read = 0 ,write = 0)
MOVLW	011	L L L L L L L L L L	MOVE literal into W (mem: read = 0,write = 1)
GOTO	100	A A A A A A A A A A	Unconditional Branch to a particular PC. PC is NOT stored in the stack (mem: read = 0 ,write = 0)
RETN	101	X X X X X X X X X X	LOAD PC stored in STACK (mem: read = 0 ,write = 0)
NOP	110	X X X X X X X X X X	No Operation (mem: read = 0 ,write = 0)
SKWCF	111	X C C X F F F F F F	SKIP instruction on W compared with F. Compare condition determined by CC value. (mem: read = 1 ,write = 0)

You have built the ALU in previous labs which takes 12 instructions as inputs (only one or zero are ever '1' simultaneously). The datapath operations that you have implemented include the 8 ALU instructions and the 4 SKWCF instructions. The remaining 6 instructions are given above as MOVF (move value between registers), CALL (call subroutine), MOVLW (move literal), GOTO (unconditional jump), RETN (return from subroutine) and NOP (no operation).

ALU and MOVF instruction format

Instruction Set - ALUWF

ALUWF	000	ADD	ADD W WITH F
ALUWF	001	SUB	SUBTRACT F FROM W
ALUWF	010	AND	AND W WITH F
ALUWF	011	OR	OR W WITH F
ALUWF	100	XOR	XOR W WITH F
ALUWF	101	INV	INVERT F
ALUWF	110	SHLTF	SHIFT F LEFT BY 1 BIT
ALUWF	111	SHRTF	SHIFT F RIGHT BY 1 BIT

All ALUWF instructions involve reading and writing to memory

Instruction Set - MOVF

'D' bit

MOVF	0	MOVWF	MOVE W TO F
MOVF	1	MOVFW	MOVE F TO W

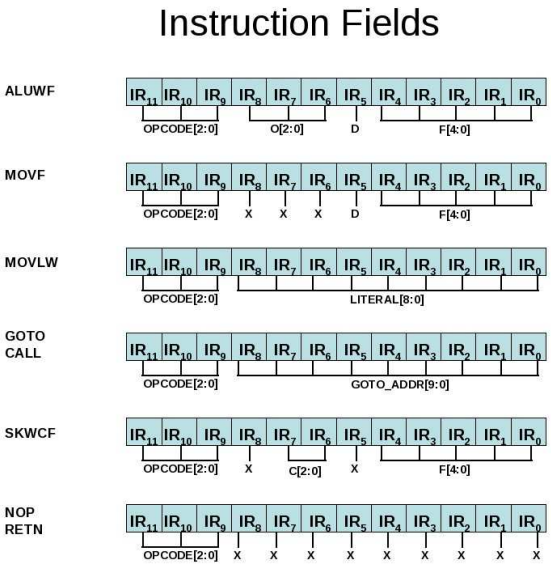
All MOVF instructions involve reading and writing to memory

SKWCF instruction format and general instruction field definitions

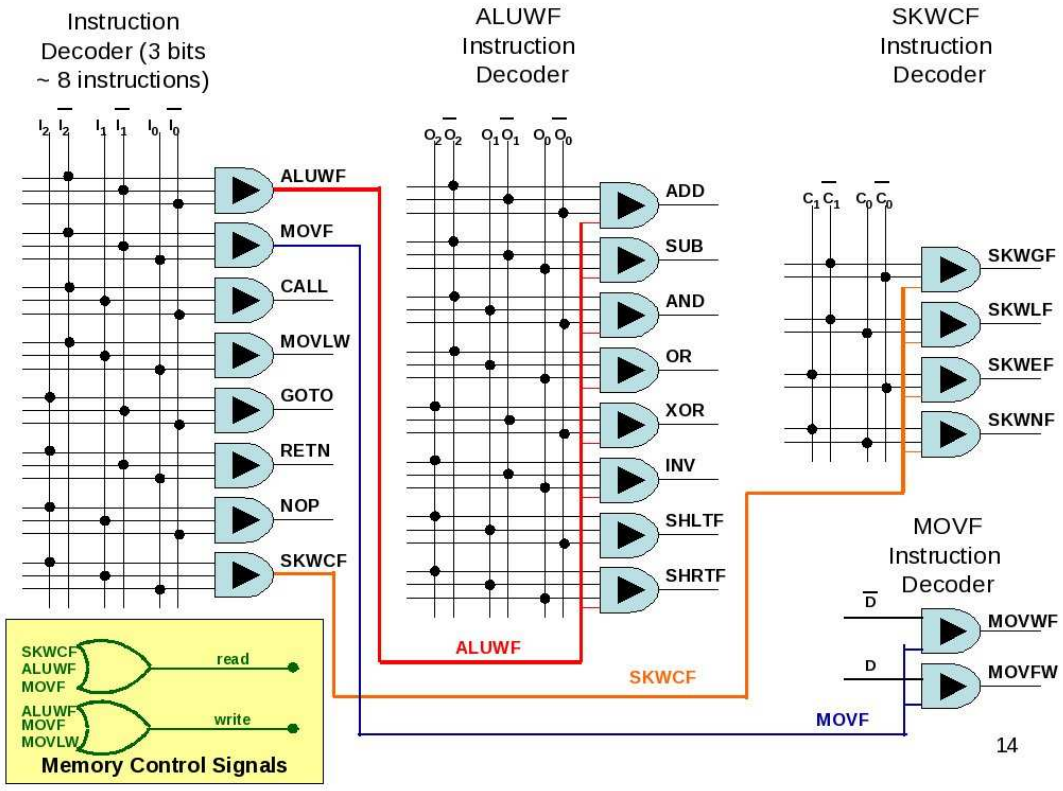
Instruction Set - SKWCF

SKWCF	00	SKWGF	SKIP IF W GREATER THAN F
SKWCF	01	SKWLF	SKIP IF W LESS THAN F
SKWCF	10	SKWEF	SKIP IF W EQUAL TO F
SKWCF	11	SKWNF	SKIP IS W NOT EQUAL TO F

Skip instructions involve only memory read
They do not involve any memory write.



Instruction decoder



The **inputs** to your instruction decoder are as follows:

opcode: 3-bits
subcode: 3-bits
d_bit: 1-bit
prog_mode: 1-bit

The **outputs** of your decoder and their function are as follows (all are 1 bit signals):

If *prog_mode* is asserted, the opcode is FORCED to NOP (110), otherwise opcode remains unchanged.

One of *aluwf_ins*, *movf_ins*, *call_ins*, *movlw_ins*, *goto_ins*, *retn_ins*, *noop_ins* or *skwcf_ins* is asserted when the opcode indicates either an ALUWF, MOVF, CALL, MOVLW, GOTO, RETN, NOP or SKWCF instruction.

One of *add_ins*, *sub_ins*, *and_ins*, *or_ins*, *xor_ins, inv_ins, shltf_ins* or *shrtf_ins* is asserted when the opcode indicates an ALUWF instruction and the subcode is one of ADD, SUB, AND, OR, XOR, INV, SHLTF, SHLRF (see table above).

One of *skwgf_ins*, *skwlf_ins*, *skwef_ins* or *skwnf_ins* is asserted when the opcode indicates a SKWCF instruction and the middle 2 bits ofthe subcode field are decoded as shown above.

One of *movwf_ins* or *movfw_ins* is asserted when the opcode indicates a MOVF instruction and the *d_bit* is a 0 or 1 respectively.

The signal *read_ctrl* is asserted when the opcode indicates an ALUWF, MOVF or SKWCF instruction.
The signal *write_ctrl* is asserted when the opcode indicates an ALUWF, MOVF or MOVLW instruction.

You should start with lab #6 and add an instruction register that is 12-bits wide. To load the instruction register using the UART, you will specify an instruction by entering a third value using the keyboard (the first two values are the A and B operands as in lab #6). You will need to modify the state machine that I've given you to add a third (instruction) register. The instruction register replaces the pushbuttons you used in lab #6 (you can eliminate these statements) and are used to control the ALU. When you press the enter button on the FPGA, the output of the ALU will be displayed in the hyperterminal.

Connect the *prog_mode* signal to the *up_button_level* signal.

Note, some of the output signals above are unused at this point, so ignore warning from the synthesis tool. In particular, *aluwf_ins*, *call_ins*, *goto_ins*, *retn_ins*, *noop_ins*, *skwcf_ins*, *movwf_ins*, *movfw_ins*, *read_ctrl* and *write_ctrl* should not be connected.

Laboratory Report Requirements:

NOTE: NEW POLICY: You will loose 10 pts/day every day the lab report is late. If you miss the demo, then the earliest you can do it is the next class period. For each class period that you are late, you will loose 20 pts.

1) Turn in a commented copy of your VHDL code along with a schematic.
2) Write a test bench and run simulation(s) that shows the inputs and output behavior of the circuit (be sure to include a set of 'sample' waveforms in your report).
3) Be prepared to give a demonstration in class on Wed using my UART driver code as described above.

Grading:
Your lab grade will consist of two parts. The first part is associated with the in-class demo, and is worth 50% of the total grade (50 pts). Successful demonstration of the lab's stated requirements is worth 50 pts. Partial implementations will be given only partial credit. The second portion of the lab grade is derived from your lab report. Correct implementation counts for 15 pts (of the 50 pts). Well documented simulation results are also worth 15 pts. The remaining 20 pts will be given according to how well the VHDL code is written and documented (comments). Bonus points will be given to any implementation feature that goes above and beyond the requirements.