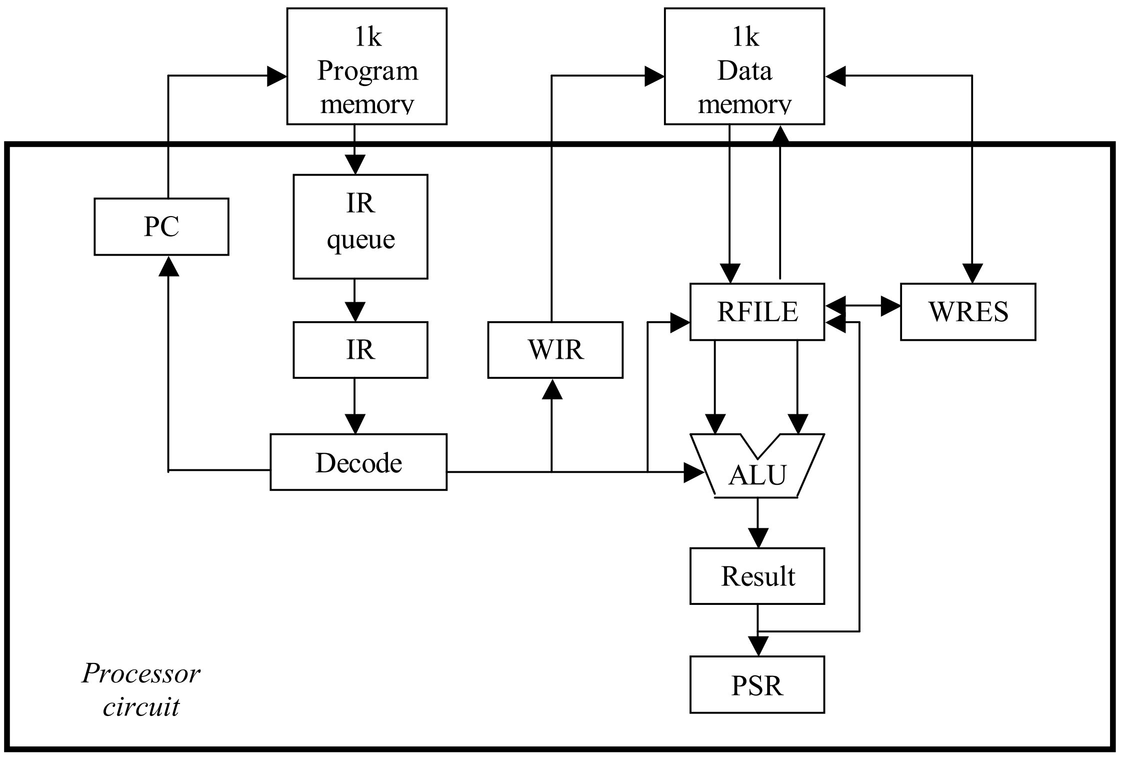
ESE355: VLSI System Design
(Miniproject)
Interim Report and Review Meeting: Friday April 23 2004
Final Report and Demos: Monday May 10 2004

Acknowledgement: The topic of this project was inspired after S. Narayan, F. Vahid, D. Gajski, "Modeling with SpecCharts", Technical Report #90-20, University of California at Irvine, 1990.

1. Objectives

The objective of the miniproject is to complete the logic level design of a pipelined processor and then realize the layout design of the corresponding VLSI circuit.

Your will have to maximize the speed of your processor. To accomplish this goal, you will have to design fast building blocks, and then carefully layout them. Your final design will have to fit inside the pad frame. *Your design should be compact, and the amount of white space in your layout should be minimized.* The I/O signals of your design will have to be connected to the I/O pads of the pad frame. The pad frame, the dimension of the pad frame and the structure of the I/O pads will be given to you by the TA.



2. The Pipelined Processor

The figure above shows the block diagram of the processor. The processor supports two address instructions, where an instruction contains a register-to-register operation, or a register-to-memory transfer (load or store). Immediate operands can also be specified.

Other processor components are a program counter (PC), an instruction register (IR), and ALU with add, compare, shift and rotate operations, status register (PSR), and an 8 register register-file (RFILE). Your processor will process only unsigned integer data of 8 bits.

An instruction is executed in three stages: fetch, execute and write result. The <u>fetch stage</u> reads the instructions pointed by the program counter into the instruction register. The <u>execute stage</u> either loads the PC with a new address (for branch instructions), loads the result register with the result of an operation (from an ALU operation instruction), or moves data between register and memory (for a load or store instruction). If the instruction was an ALU operation, then the <u>write results stage</u> writes the contents of the result register to the appropriate register or memory location.

Instructions in IR register are decoded through the decode unit, and the proper control signals are generated for the ALU, PC, register file, Result register, program and data memory.

We assume that instructions are kept in a 1k instruction memory, and data are read/written into a 1k data memory. Thus, when you design the instruction word you have to make sure that your processor can address 1k of memory space. The content of the PC register is used as an address to access the instruction memory. Similarly, the content of register WIR is the address for the data memory. The data involved in a memory transfer will be read from or written into register WRES.

To increase the efficiency of program execution, an instruction pipeline will be implemented. The fetch, execute and write stages all perform in parallel in a single clock. Fetch stage fetches an instruction into the instruction queue (IR queue). Execute stage pops an instruction off the instruction queue into IR register and then executes the instruction. If there is no available instruction in IR queue then execution is idle. Write results stage stores the result of an instruction from the previous clock cycle to the appropriate memory location. Note that the destination address will be in register WIR and the data will be in register WRES. We assume that memory access (for read or write) is done in a single clock cycle.

The instruction set will include following instructions:

(a) LD Reg, memory address – load instruction transfers value from the memory address to register Reg in the register file
(b) LD Reg1, Reg2 – load instruction that transfers the contend of Reg2 into Reg1
(c) LD Reg, PC – loads the content of register Reg into the PC counter
(d) LD PC, Reg – loads the content of the PC counter into register Reg of the register file
(e) ADD Reg1, Reg2 – adds the content of registers Reg1 and Reg2 and stores the result back to register Reg1, as a result OV (overflow) and Z (zero) flags are set
(f) CMPE Reg1, Reg2 – checks if Reg1 = Reg2, as a result a certain EQ flag is set in register PSR
(g) CMPL Reg1, Reg2 – checks if Reg1 < Reg2, as a result a certain LT flag is set in register PSR
(h) SHF Reg – shift left the content of register Reg
(i) BRZ address – branch to address if Z flag was set by the previous instruction
(j) BRE address – branch to address if EQ flag was set by the previous instruction
(k) BRL address – branch to address if LT flag was set by the previous instruction
(l) BROV address – branch to address if IV flag was set by the previous instruction
(m) ST Reg, memory address stores the content of the Register into the memory location

3. Logic Level Design

To complete your processor design, you will have to go through following steps:

1) Decide the structure of the instruction word.
2) Encode the opcodes of the 13 instructions of your processor.

3) Decide the control circuits that must be generated to the processor blocks after decoding an instruction.
4) Decide the handshaking signals between the processor and the 2 memory modules.
5) Design the register file including the decoding circuitry.
6) Design the IR queue.
7) Design arithmetic modules for addition, comparison, shift and rotation.
8) Design the decoder circuit.
9) Design the other registers, like PC, IR, PSR, WRES, WIR, Result.
10) Design the processor FSM which coordinates the pipelined execution of the fetch, execute and store phases.
11) Predict the processor speed.

4. Testing of the Processor

You will have to decide on the testing procedure for your processor chip. This includes the testing of your logic design, as well as testing after layout design and extraction.

In your reports, you should indicate what test patterns (test programs) you are using. Explain the reason behind choosing a certain set of test patterns.

Explain what additional circuitry you introduced to simplify the testing procedure. A testing scheme that is commonly used is *scan chaining*. This includes having your flip flop outputs sent to pads so that you can test what is going through them at any point in time.

5. VLSI Implementation

You are advised to follow the following guidelines:

- The design will be implemented in TSMC 1.0um technology.
- You will use the same standard cell library like the one you used for Homeworks 3 and 4.
- If possible, try to follow a modular design style in which several circuits form a module, which could be used for another design also. For example, the adder, shift and rotate circuitry should for ALU module. The control logic of your FSM can be another module, and so on.
- To decrease the design effort, use the bit-sliced design style whenever possible.
- Minimize the wasted area. Circuits should be placed in a compact rectangular area, which has a minimum amount of white (unused) space inside. A lot of white spice indicates that you did not carefully layout your design, so at the end, your circuit will cost more than it should.
- Avoid long and parallel wires. They will introduce large routing capacitances, thus contribute to decreasing the speed of your processor.
- Carefully design global wires like clock, VDD and GND. If needed, buffer the global wires.

6. Simulation

You should submit simulation results for (1) logic simulation, (2) irsim simulation of the transistor level design and (3-4) simulation after layout extraction (with and without pad frame).

7. Interim Report and Review Meeting

The design review is intended to make the flow of the design cycle a little easier. Designing of larger circuits might become very difficult, if not managed well.

Each team will sign-up for a time slot, and present to me and to the TA - the way they plan to design the pipelined processor. You will get a grade, which is 40% of the final project grade.

For the interim report and the review meeting, you should complete (1) the logic level design and decide the (2) testing procedure of your processor. (3) You should have decided the I/O signals between the processor and the two memory modules. (4) The report should include the appropriate simulation waveforms to motivate that the logic design is correct.

8. Final Report

The final report should have the following structure:

- *Cover page*: project name, name, date, email addresses and full path to the cadlab directory where the Magic files can be found
- *Project description*.
- *Description of the logic design*, including all blocks, FSMs, pipelining, communication protocol with memory. Explain the expected processor speed.
- *Description of the testing procedure*: test patterns/test programs, test logic, and so on.
- *Description of the VLSI design*, including layouts of blocks, modules and the entire chip, Pin diagram and explanation how each pin is used.
- *Experiments*. Output waveforms for your processor for the four simulation kinds.
- *Performance of your design*: characterize the speed, area, and testability of your processor.
- *Conclusion*. Conclude your miniproject experience.