

- Modify the existing overall CPU architecture to avoid structural, data and control-type hazards in RISC CPUs. Observe data dependencies among instructions, CPU stalls due to branches and jumps. Learn the use of multi-port memories, forwarding loops and stall circuitry.
- Understand and solve the data hazards due to multi-cycle ALU operations and superscalar RISC architecture.
- Be able to design the CPU control circuitry that controls a simple CPU pipeline with forwarding loops, and other hazard-preventing circuitry.
- Floating point unit and its pipeline architecture.
- Understand SRAM topology, architecture, circuitry and control.
- Understand the principles behind cache memories, cache types, topology and trade-off. Be able to architect a direct-mapped, set-associative or full-associative cache.
- Understand cache-read and write mechanisms and the circuitry. Be able to describe the difference between a write-through and a write-back cache in detail, understand each operation.
- Understand the operation of a simple cache controller.

Course Schedule

Date	Topics
WEEK 1	Introduction to timing, definitions of set-up and hold time for flip-flops.
WEEK 2	Complex system timing and pipelining issues using flip-flops, timing violations.
WEEK 3	Principles of computers, addressing modes, RISC instruction format.
WEEK 4	Basic RISC instruction set.
WEEK 5	Implementation of the 32-bit RISC scalar architecture.
WEEK 6	Continue on implementation of the RISC scalar architecture.
WEEK 7	Pipelining hazards: Structural hazards.
WEEK 8	Pipelining hazards: Data hazards.
WEEK 9	Midterm exam. Continue on Data hazards.
WEEK 10	Pipelining hazards: Control hazards.
WEEK 11	Multi-cycle operations in pipelines. Superscalar RISC CPUs. Floating point unit.
WEEK 12	Continue on Floating point unit.
WEEK 13	Principles of memory architectures. Register File and Caches.
WEEK 14	Continue on memory architectures.
WEEK 15	Continue on memory architectures.
WEEK 16	Final exam.

Laboratory Schedule

All laboratories are conducted in Mentor Graphics ModelSim environment and ALTERA DE2 FPGA boards using Quartus II environment.

- A D-type flip-flop, 1-bit register and 8-bit shift register.
- A simple controller design (Moore machine).
- A 32x16 SRAM implementation.
- Memory-to-memory transfer, non-pipelined logic circuits.
- A pipelined RISC CPU design.

Organization of Laboratory Reports

OBJECTIVE

- Mention the name of the experiment or the circuit you are building here.

DESCRIPTION

- Describe the experiment or the circuit here.

CIRCUIT DIAGRAM

- Draw the circuit diagram(s), truth table(s), state diagram(s), etc. here.

VERILOG CODE

- Include the Verilog code you have saved during the experiment here.

VERIFICATION/TEST STRATEGY

- Include a detailed description of the functional verification test strategy here. How do you plan to test the circuit you have built for the experiment?
- If the circuit is combinatorial, the test plan is usually its truth table. If it is a state machine the state diagram will lead you to put together a test strategy for it.

RESULTS

- The waveforms you saved during the experiment must be included here.

CONCLUSIONS

- Describe what you have learned from this experiment here.

Policy on Cheating

A student or students involved in a cheating incident involving any non-exam instrument (homework, report, or lab project) will receive an F on that instrument, and will be reported to the judicial affairs office. Whether the report will carry a recommendation for disciplinary action will be left to my judgment.

A student or students involved in a cheating incident on any quick test, the midterm exam or the final exam will receive an F in the *course*, and will be reported to the judicial affairs office *with* a recommendation for disciplinary action.

(see http://www.sjsu.edu/student_affairs/academicdishonestyrevisedpolicy.pdf)