Design and Characterization of a CMOS 8-bit Microprocessor Data Path

Project Summary

This project involves the schematic and layout design of an 8-bit microprocessor data path, including an ALU, a barrel shifter, and a register file (SRAM) using CMOS circuitry and Cadence VLSI design tools. The project will incorporate all the skills and knowledge students have gained from the individual lab assignments and provide a team of students with an opportunity to work on a more open-ended digital integrated circuits design problem.

Data Path Overview

The data path is the core element of a microprocessor that can perform many of functions, including arithmetic and logic evaluation, data movement, and storage of results in a memory address specified by the instruction. In this design project, teams must design a complete 8-bit data path consisting of a register file (SRAM), an ALU (arithmetic logic unit) and a shifter, as shown in *Figure 1*. The data path will be capable of multiple instructions determined by several control signals that comprise the operational code or "*opcode*".

The ALU can perform both arithmetic and bit-wise logical operations. The ALU will have two 8-bit data inputs and several control signals that specify the ALU function to be executed in a given clock cycle. The ALU input data is loaded into the latch at the rising edge of the ALU clock (*clk2*), and the output is generated after a delay due propagation through the combinational logic of the ALU and the shifter. The central component of an ALU is an adder, and for this project you are expected to select and justify the best 8-bit adder that will be used.

A shifter provides data manipulation capabilities. In this design project, a barrel shifter will be used because it has a very efficient layout and can perform n-bit shifts in a single clock cycle.

The register file is a block of memory that provides the data inputs to and stores outputs from the ALU. The register file at read/write address is specified by control bits included in the opcode. In this project, the register file and ALU/shifter will be synchronized by two non-overlapping clock phases that determine when data is latched into the register file (clk1) and into the ALU (clk2).

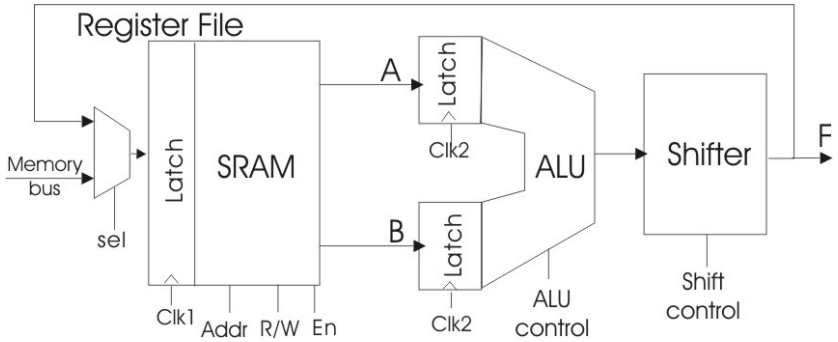

Fig. 1: Structure of the microprocessor data path to be designed in this project

Data Path Design Specifications

The following specifications must be achieved by the data path you design

- bit width: 8
- inputs: one 8-bit words (from the memory bus), 16 control/address bits (7 for the
- ALU/shifter, 11 for the register file), two non-overlapping clocks (*clk1*, *clk2*).
- outputs: 8-bit word (F), 1 carry-out
- functions: must implement at least the following functions:

  ALU functions

  - A NOR B
  - A XOR B
  - Increment A
  - Decrement A
  - A NAND B
  - A + B
  - A - B
  - NOT A (invert)

  If you implement only these functions, do so with *only three* ALU control signals. There is no explicit Cin bit; Cin must be generated from the ALU control bits.

  Shifter

  The shifter must implement

  - *o* Pass (no shift)
  - *o* Shift_Left_1
  - *o* Shift_Left_2
  - *o* Shift_Right_1
  - *o* Shift_Right_2

  Expanded functions

  Two function expansion bits are provided to implement additional ALU and/or shifter functions (as a bonus)

  Register file

  The 8x8 SRAM register file must implement 1-port *read_from_memory* and *write_to_memory* functions. Each clock cycle will be either a read or a write cycle, so two cycles will be needed to complete a function.

  - ***power supply***: VDD = 1.8 volts, referenced to 0 volt ground.
  - ***size specifications***: no specific layout size is required, but optimization of physical size is a design priority and higher scores will be awarded to groups with smaller layouts. The floorplan in *Fig. 2* will assist in achieving an optimal layout.
  - ***timing specifications***: signal timing requirements are defined below. The worst case propagation delay for all ALU+shifter functions must be less than **5ns**

Control Bit Specifications

This design project must be completed with only the following 16 control signals: 1-bit register file input mux selection, two 3-bit register addresses, 1-bit register file read/write, 1-bit register file enable, 3- or 4-bit ALU controls, and 3-bits shift controls. These control signals must be given the signal names defined below. Note: this only defines the externally-supplied control signals; you can decode or otherwise manipulate them to construct any necessary internal control signals.

Data Path Layout

The 8-bit datapath can be implemented by constructing a 1-bit slice horizontally and stacking 8 one-bit slices vertically. This layout floorplan for the data path is shown in *Fig. 2*. The data flows horizontally from the left to the right. The control signals run vertically across each slice. You may implement some functional blocks as 8 bits, but make sure you organize it in 1-bit slices that can be pitch-matched to the other functional blocks

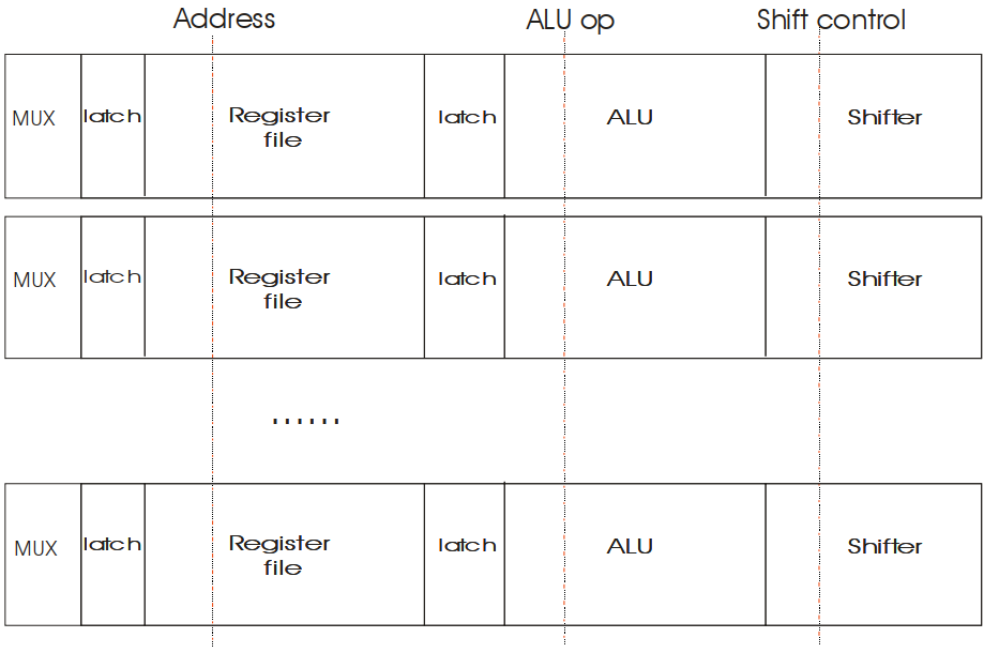

Fig. 2: Floorplan of the data path

Register File Description

The 8-bit register file should be implemented as an 8x8 SRAM. Since the SRAM is small, you do not have to implement a sense amplifier. The register file should have two 8-bit read (output) ports and one 8-bit write (input) port. Input to the register file can come either from the ALU output or from an external memory bus, as selected by a MUX at the input of the register file. The memory bus should be implemented as an 8-bit input word. An 8-bit latch should be included to store the register file input word, which, during a write cycle, will be saved to the SRAM block.

The register file uses an enable signal for the general task of enabling inputs to and outputs from the register file at the proper time. The register file must be carefully designed to properly implement the enable function. In the read cycle, register file output is passed to the ALU when enable is high; otherwise that output should be at high impedance. The output should also be at high impedance during the entire write cycle, when the R/W signal is high. In the write cycle, data is stored in the SRAM only when enable is high; otherwise the data from the register file latch should not be allowed to affect the SRAM. To accomplish this, the enable signal should act on the SRAM address decoder outputs. The enable function allows the data path to implement functions that do not act on the register file, such as branches or status register checks, although these functions are not implemented in this design project.

The register file address actually contains two independent 3-bit addresses. Address A should specify the address for data passed to ALU input A, and address B should specify ALU input B. Address A is also used to specify the address for storing data to the SRAM during a write cycle. The timing of the data path should allow the value of address A to be changed before the write cycle, as shown in Fig. 3. However, for this project you can keep the address constant and store the ALU/shifter data back to the same location where input A was taken from.

Clock/Timing Description

Two non-overlapping clock phases are required for the data path, as shown in *Fig. 3*. Data from the ALU/shifter or memory is loaded in the register file latch when clk1 is high and held there while clk1 is low. Data from the register file is loaded in the ALU latch when clk2 is high and held there when clk2 is low. Two cycles of the clocks are needed to complete a data path function; data is passed to and through the ALU/shifter in the "read" cycle, and ALU/shifter data is stored in the register file on the "write" cycle. Address and control signals for all data path functions must be synchronized to the clock signals specifically as shown in *Fig. 3*.

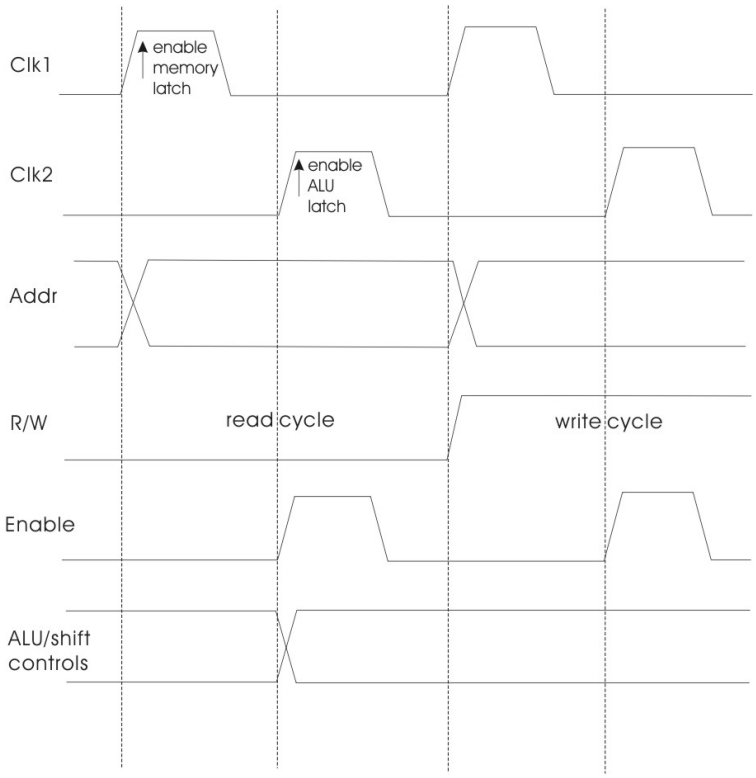

Fig. 3. Timing diagram for a complete 2-cycle data path function. The first cycle is a register file read to pass date from the register file to the ALU. The second cycle is a write to store ALU/shifter outputs to the register file.

Project Requirements

The following cell views for all circuit blocks must be created using Cadence tools and stored in your group project directory:

- schematic
- symbol
- layout

The complete data path cell must pass the following tests and verifications:

- functional simulation of all data path functions
- Timing analysis

Input and output signals must be named as follows:

- Inputs
  - md<0:7>, memory bus data
  - rfs, reg. file mux selection
  - rfen, reg. file enable
  - rw, read/write
  - ada<0:2>, register address A
  - adb<0:2>, register address B
  - f<0:2>, shifter controls
  - f<3:5>, ALU controls
  - f<6>, expanded function control
  - clk1, clk2
- Outputs
  - Y<0:7>
  - Cout

The following characteristics must be measured and reported:

- slowest logic function and propagation delay for that function (ALU)
- slowest arithmetic function and propagation delay from that function (ALU)
- slowest propagation delay of the data path (ALU + Shifter)
  Propagation delays are measured from clock edge to last output transition.
- physical area of the complete data path layout
- total number of transistors in the data path (available from the final LVS output)

Circuit Requirements:

- an 8-bit adder must be implemented
- a barrel shifter must be implemented
- the register file memory must be implemented with a multi-port 8x8 SRAM
- minimum sized transistors can be used, but are not required
- when possible, existing primitive cells (INV, NAND, etc.) should be used to construct circuit blocks, and any new primitive cells (such as the carry generation circuits) must be designed at the transistor level
- all required functions must be implemented using only the specified control/select inputs
- you must employ hierarchical design where you instantiate lower level cells into higher level circuit block, both in schematic and layout design
- buffers should be designed and included wherever fan-out is larger than 5 gates

Layout Requirements:

- follow all previously established layout guidelines for cell size (pitch), internal routing, input/output pad placement, etc.
- all designed circuit schematics must be laid out
- all functional blocks should be designed using only *metal1* and *metal2*, reserving *metal3* for routing chip-level control and data signals in the top level of hierarchy
- all cell I/O signals and power traces must be connected to single points in the final cell, i.e., you can't have multiple ports for *VDD*, *Ground*, or any I/O signal
- attempt to consistently route *metal1* and *metal2* in perpendicular directions. you may need to twist cells around and stack them in seemingly odd ways to achieve your goals, but always make sure you have good power supply to all cells and that you have good access to run *metal2* and *metal3* in a single direction in your final data path cell

Project Planning and Considerations

- Your group should decide during the proposal stage exactly what cells you will need to complete your project. Although you can alter this plan slightly if necessary, you should identify early on what functions you will construct at the 1-bit level, at the 4-bit level, and at the full 8-bit level.
- Your group must decide which tasks of this design project will be completed for each design phase. Since layout and final simulation/characterization take much more time than schematic capture and functional simulation, you must get beyond the schematic stage by **January 21**.

Deliverables:

All schematic and layout cells required to construct the data path must be within a single user account directory. Simulations should be preformed for verify ALL data path functions, several of which will be tested during the project demonstration. In addition, the following must be provided within the final project report

1	select schematics that demonstrate your design effort
2	final data path schematic and symbol
3	a truth table of control inputs and ALU/shifter functions
4	functional simulations, including loading data from memory, perform XOR, A+B, and A-B operations on the data, and saving the result to another register in the register file.
5	final data path layout and any smaller cell layouts that you feel you did an exceptional job optimizing
7	post layout simulations showing propagation delays as specified above
8	table of cell specifications including - measured cell width, height, and area, number of transistors, and total area/transistor - measured performance characteristics (worst-case delays and power)