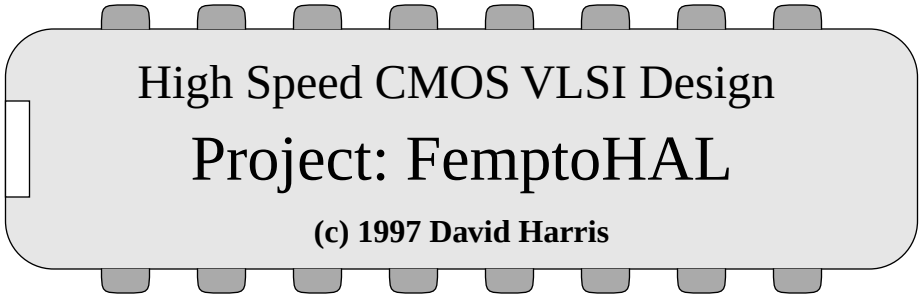



# 1.0 Introduction

The objective of this project is to design the integer execution core of a blindingly fast 8-bit microprocessor! This processor will be known as the FemptoHAL. The project should tie together many of the ideas you are learning, including skew-tolerant domino design, sizing of realistic paths, and high-performance pipelining.

For this project, you should work in a team of two. There will be a prize for the team implementing the fastest project. The exact prize has not yet been determined, but will either be an exotic vacation to Hawaii or a nice dinner for two.

## 1.1 Schedule

**10/22-23:** Verilog Model of FemptoHAL

**10/28-29:** Initial design of add/subtracter (prize for speed!)

**11/3-11/4:** Initial design of register file

**11/5-11/6:** Complete schematics of FemptoHAL

**11/10-11/11:** Present optimized FemptoHAL implementation

More details on exactly what must be handed in will be available as the class continues.

# 2.0 Architecture Specification

The Instruction Set Architecture (ISA) for the FemptoHAL is extremely simple, consisting of only integer instructions. Since you will not be implementing memory or branch sub-

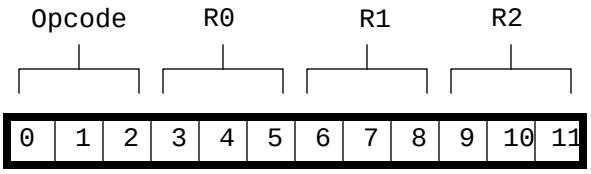
systems, such instructions are not included. The instructions your processor must handle are:

**TABLE 1. Instruction Set**

| Instruction | OPCODE | Notes |
|---|---|---|
| LDC R0, CONST | 000 | R0 <- CONST (an immediate constant) |
| ADD R0, R1, R2 | 001 | R0 <- R1 + R2 |
| SUB R0, R1, R2 | 010 | R0 <- R1 - R2 |
| AND R0, R1, R2 | 011 | R0 <- R1 AND R2 |
| OR R0, R1, R2 | 100 | R0 <- R1 OR R2 |
| NOT R0, R1 | 101 | R0 <- NOT R1 |
| XOR R0, R1, R2 | 110 | R0 <- R1 XOR R2 |
| NOP | 111 | No OPeration |

The architecture specifies eight general purpose 8-bit registers. Each instruction uses a 12 bit encoding, in which the first three bits are the opcode, the second specify R0, the third specify R1, and the fourth specify R2, as shown in Figure 1. For LDC instructions, eight more bits containing a constant are automagically fetched.

**FIGURE 1. Instruction Encoding**




# 3.0 Microarchitecture Specification

## 3.1 Pipeline Overview

The microarchitecture of the FemptoHAL is a single-issue, in-ordered pipelined machine. The pipestages are:

- FETCH: Fetch the instruction and optional 8-bit constant
- REG: Access the register file or bypass result
- EXE: Execute instruction
- WRB: Update register file with new result

To prevent pipeline stalls, instructions must be able to bypass their result to dependent instructions later in the program before writeback completes. For instance, consider the following program:

```
LDC R0, #1
```

```
ADD R1, R0, R0
ADD R2, R1, R0
ADD R3, R2, R0
```

The results of previous instructions are bypassed to become sources of later instructions.

## 3.2 Inputs

The inputs of your pipeline are listed below. The pipeline has no outputs! Each signal name ends with the letter F, R, E, or W, indicating that it becomes valid in the FETCH, REG, EXE, or WRB pipestage, respectively.

**TABLE 2. Inputs**

| Signal | Purpose |
|---|---|
| Src1AdrF[2:0] | Register ID of Source 1 |
| Src2AdrF[2:0] | Register ID of Source 2 |
| DstAdrF[2:0] | Register ID of Destination |
| DstVldF | Destination register ID valid (0 for NOP) |
| ConstBarF[7:0] | Value of constant, inverted (for LDC operation) |
| ConstSelR | Select SRC1 = constant (for LDC operation) |
| OpR[4:0] | Operation (00001 = Add/Sub, 00010 = AND, 00100 = OR, 01000 = XOR, 10000 = NOT/LDC) |
| AddCtlR[1:0] | Adder Control signal (bit 0 = Cin, bit 1 = subtract) |
| ph1-ph4, php | Four clock phases + pulsed clock |

A decoder that you do not need to implement produces these control signals from the fetched opcode. LDC is performed by a NOT operation on the complementary version of the constant. ADD and SUB share the same 8 bit adder/subtracter using the AddCtlR control signal to optionally invert the input and supply a carry in for subtraction. NOP is performed by setting DstVldF to 0, indicating an invalid destination address which should not be matched by the Bypass Control logic or written back to the register file.

All inputs are available 150 ps before the end of the appropriate cycle, except ConstSelR, which becomes valid 100 ps after the start of the REG stage. Inputs come from non-monotonic static logic.

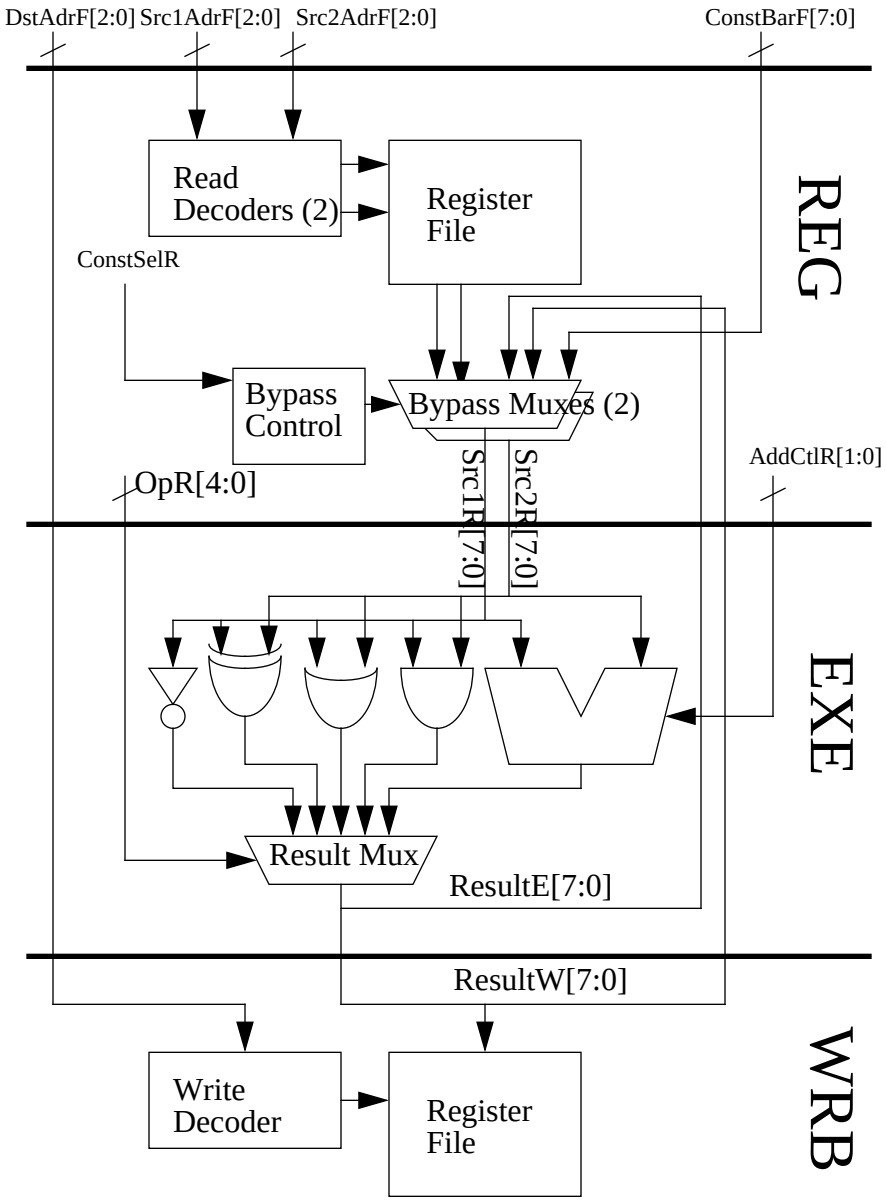
Although a realistic design would support scan, you already have enough work to do and do not need to implement scan.

## 3.3 Pipeline Diagram

Figure 2 shows a pipeline diagram of the FemptoHAL processor microarchitecture that you are responsible for implementing. Note that no logic optimizations have been done;

for example, the 5-input result multiplexor might be faster if implemented with two stages of smaller multiplexors. You are free to optimize the pipeline but may not add pipe stages.

**FIGURE 2. FemptoHAL Pipeline Diagram**




The bypass control block takes source and destination addresses from the various pipe stages to control the bypass multiplexors to select either the current value read from the register file, a constant, or results of previous instructions in the EXE or WRB stages. These signals are not shown, so you will have to do some design. DstVldF is also not shown, but must be used by the write decoder and bypass control.

# 4.0 Block Specification

## 4.1 Function

Your circuits should implement the microarchitecture described in the previous section. You may use any circuit tricks you want as long as they do not significantly degrade yield. You may also make simple modifications to the microarchitecture, but still must correctly implement the architecture and should not do anything that could reduce IPC such as add more pipeline stages or remove bypass paths. Check with the instructor if you have creative ideas that might violate these guidelines.

Plan to use skew-tolerant domino circuits wherever necessary for high speed. Although wire capacitance is important, most wires in this chip will be fairly short, so neglect them for simplicity.

## 4.2 Delay

The cycle time of your machine should be as fast as possible! You should be able to do substantially better than 1 GHz.

## 4.3 Input and Output Capacitance Limits

You may assume that all inputs come from 1 mm wires and may present loads on the inputs comparable to the wire capacitance if necessary.

The circuit produces no outputs because there are no STORE instructions in the limited architecture. The sizing of the circuit will be set by requirements that it bypass to itself. Thus, you could make the entire circuit ten times larger than minimum and run at the same speed. Resist the temptation.

## 4.4 Area & Power

You may spend as much area and power as necessary to run extremely fast, but no more. Do not be inefficient by oversizing non-critical gates or by making the entire path larger than necessary.

## 4.5 Simulation

Simulate your design at the same temperature and voltage you used when measuring the FO4 inverter delay in Jobwork 3. Assume TT processing.

Producing clocks will be difficult since your cycle times will be so short. To keep the project manageable, you can assume ideal clocks. Your block will receive four domino phases ph1-ph4 with 50% duty cycles, spaced by 90 degrees. It also receives a pulse php with the rising edge aligned with the first domino phase and a pulse width (50%-50%) of 3

FO4 delays. All clock edges have 20-80% rise/fall times equal to the rise/fall time of a FO4 inverter.