ENCM 369 Winter 2023 Lab 8 for the Week of March 13

Steve Norman
Department of Electrical & Software Engineering
University of Calgary

March 2023

Administrative details

You may work with a partner on this assignment

If you choose to work with a partner, please make sure that both partners fully understand all parts of your assignment submission, and please follow the instructions regarding submission of your PDF document.

Partners must be in the same lab section. The reason for this rule to keep teaching assistant workloads balanced and to make it as easy as possible for teaching assistants to record marks.

Due Dates

The due dates don't follow the usual pattern, for two reasons. First, it seems fair to allow extra time after Midterm #2. Second, I was late to post this document.

The Due Date for this assignment is 6:00pm Thursday, March 23.
The Late Due Date is 6:00pm Friday, March 24.

The penalty for handing in an assignment after the Due Date but before the Late Due Date is 3 marks. In other words, X/Y becomes (X–3)/Y if the assignment is late. There will be no credit for assignments turned in after the Late Due Date; they will not be marked.

How to package and hand in your assignments

You must submit your work as a *single PDF file* to the D2L dropbox that will be set up for this assignment. The dropbox will be configured to accept only file per student, but you may replace that file as many times as you like before the due date.

See the Lab 1 instructions for more information about preparing and uploading your PDF file.

Important update for those working with a partner: Please submit only one PDF file for both students. On the cover page, put lab section, name and ICID information in this format:

> Group Submission for [lab section]
>
> Submitted by: [submitter's name]
> UCID: [submitter's UCID]
>
> Partner: [partner's name]
> UCID: [partner's UCID]

Marking scheme

A	3 marks
B	5 marks
C	6 marks
D	3 marks
TOTAL	17 marks

Notes about timing in digital logic

(The material from here to the beginning of the Exercise A instructions is mostly a review of material taught in ENEL 353 every year for the past several years. I originally wrote the material for students who had taken a version of ENEL 353 that had no coverage of timing. I've included the material in the 2023 Lab 8 instructions because it may serve as a short and convenient review.)

For desktop computers available in 2023, a typical processor clock frequency is approximately 3 GHz, that is, 3×10^9 cycles per second. The period for that clock frequency is 1 s / (3×10^9), so, 0.333 nanoseconds, or 333 picoseconds. That's just an astonishingly short period of time; it's hard for us humans to comprehend how short it really is.[1]

However, it is still interesting to ask why that clock period can't be squeezed from 333 picoseconds down to 200 picoseconds, or 100 picoseconds. There are two main reasons:

- For a given synchronous logic circuit, increasing the clock frequency increases power consumption. If you make the clock frequency too high, the circuit will destroy itself with its own waste heat. We won't look at power consumption in ENCM 369, but please be aware that it is an important and interesting problem.
- Circuit elements such as combinational gates and D flip-flops are *not infinitely fast*. For all of them there are very short *delays* between changes to inputs and responses of outputs.

This section reviews simple models of delays for combinational logic and D flip-flops; these models should help with understanding limits on clock speeds for the processor designs of Chapter 7 of the course textbook.

The models and terminology for delays are taken from Sections 2.9.1 and 3.5.2 of the course textbook.

Delays in combinational logic

For even the simplest combinational element, an inverter, there is a *range* of possible delays. Response of the output to a change in input depends on various factors, such as ...

- *temperature*—switching in a circuit tends to be slower when the circuit is warm than when it is cold;
- *asymmetry*—because of transistor physics, a CMOS[2] inverter might make a 1-to-0 output transition faster than it can make a 0-to-1 transition;

[1] An Olympic sprinter is considered to have false-started if he or she reacts to the starting gun in less than 0.1 seconds; for a 3 GHz clock, 300,000,000 clock cycles happen in that 0.1-second interval.

[2] CMOS stand for Complementary MOS, and MOS stands for metal-oxide-semiconductor, a kind of transistor. (Confusingly, modern MOS transistors do not contain any metal!) CMOS logic has been the dominant technology for digital integrated circuits for over three decades.

Figure 1: Contamination delay and propagation delay for an inverter. The waveform for x shows the two possible transitions for x. The waveform for y can be thought of as showing a *collection* of responses: the fastest possible, the slowest possible, and some in-between.

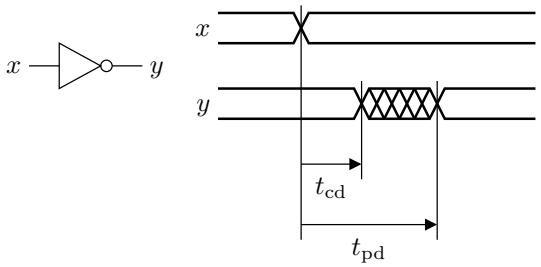
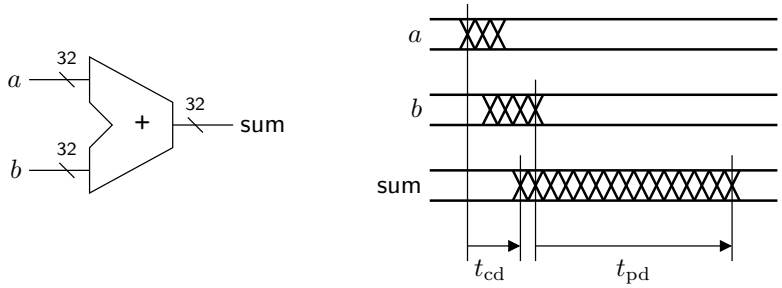


Figure 2: Contamination delay and propagation delay for a 32-bit adder. Of course, the 64 input bits are not required to change all at the same time. Due to the carry logic within the adder, changes to the 32 output bits may be quite spread out in time.



- *manufacturing variations*—two supposedly identical gates within the same integrated circuit might switch at different speeds due to imperfections in the material and equipment used to make the circuit;
- *variations in load*—if the output drives only one input of another gate, it will switch faster than it would driving many gate inputs. (Details of loads and a related concept called *fanout* are topics for third year in Electrical Engineering.)

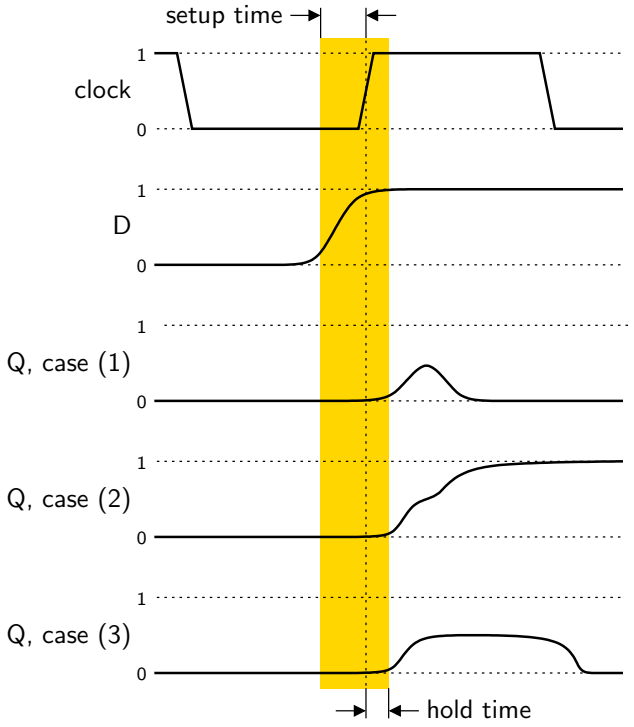
To keep things simple, it makes sense to come up with two numbers that describe the shortest and longest delays possible in response to a change of input to a combinational element. We will call these t_{cd}, *contamination delay*, and t_{pd}, *propagation delay*.

For a component with multiple inputs and multiple outputs, we need to be careful with our definitions:

- t_{cd} is the *shortest* possible delay between a change in *any single* input bit and a change in *any single* output bit.
- t_{pd} is the *longest* possible delay between the time when *all* input bits are stable and the time when *all* output bits have reached final values.

These definitions are illustrated for a 32-bit adder in Figure 2.

Figure 3: Depiction of setup and hold times for a positive-edge-triggered D flip-flop. The example D input breaks the rule that D must be constant within the sampling window defined by the setup and hold times. The given Q signals are sketches of three of the many possible outcomes: (1) Q starts to move from 0 to 1, but falls back to 0; (2) Q gets to 1, possibly more slowly than a circuit designer would like; (3) Q is "stuck" about halfway between the voltages for 0 and 1 for a significant period of time. Case (3) is an example of an undesirable phenomenon called *metastability*.



A timing model for a D flip-flop

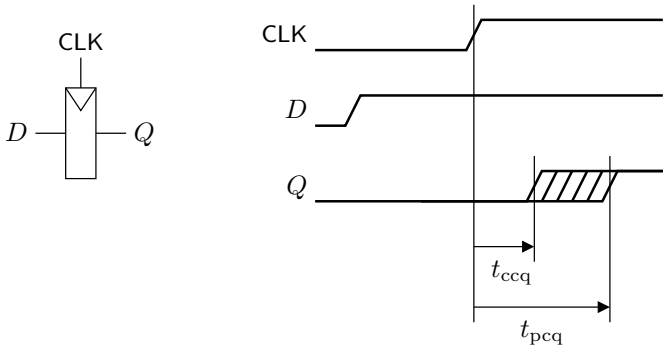
For a D flip-flop, there are two main timing concerns:

- How close in time to an active clock edge can the D input change without causing the "Q copies D on an active clock edge" behaviour to become unreliable? This is usually expressed in terms of two parameters: the *setup time*, t_{setup}, and the *hold time*, t_{hold}.
- How long after an active clock edge will it take for the Q output to change? The parameters for this are t_{ccq}—*contamination delay from clock to Q*, and t_{pcq}—*propagation delay from clock to Q*.

Setup and hold times: It's not possible to design a circuit to reliably capture the value of a D input on an active clock edge if that D input is not constant for at least some tiny time interval nearby to that active clock edge. t_{setup} is the time interval before an active clock edge for which D must be constant, and t_{hold} is the time interval after an active clock edge for which D must be constant. Figure 3 shows some of the things that can go wrong if setup and hold time constraints are not satisfied.

Clock-to-Q contamination delay and propagation delay: Informally, for a D flip-flop, we say, "Q samples D on each active clock edge, and holds that sample value until the next active edge." However, Q can't actually change state instantly,

Figure 4: Clock-to-Q contamination delay and propagation delay, illustrated for a positive-edge-triggered D flip-flop with Q changing from 0 to 1. It is assumed here that D satisfies the setup and hold time constraints for the flip-flop. The waveform for Q can be thought of as a collection of responses: fastest possible, some slower ones, and the slowest possible.



so if Q changes from 0 to 1 or 1 to 0, that change occurs with a delay that is at least t_{ccq} and at most t_{pcq}, as shown in Figure 4.

Exercise A: Review of setup- and hold-time constraints

Read This First

This exercise reviews important material from ENEL 353 about timing in digital logic circuits. For help with review of that material, you may wish to carefully read the above notes on timing of digital logic and Sections 2.9.1, 3.5.1, and 3.5.2 in the course textbook.

What to Do, Part I: Setup-time constraint

Consider the circuit of Figure 5. Suppose that for the registers t_{pcq} is 27 ps and t_{setup} is 32 ps. Suppose that for the sign-extension unit t_{pd} is 38 ps, and for the multiplier t_{pd} is 195 ps.

If the desired frequency for CLK is 2.80 GHz, what is the maximum acceptable t_{pd} for the adder?

What to Do, Part II: Hold-time constraint

Again consider the circuit of Figure 5. If t_{hold} for the registers is 7 ps, what is the minimum acceptable t_{ccq} value for the registers?

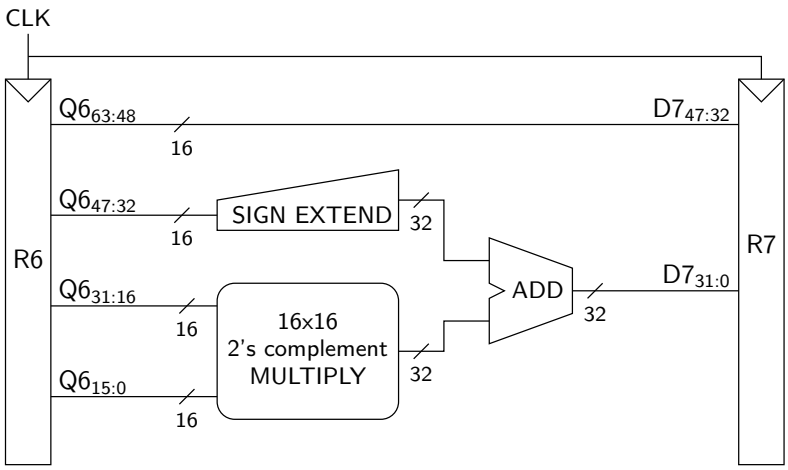
What to Include in Your PDF Submission

Include clear and precise calculations of you answers to Parts I and II.

Exercise B: Propagation delay in processor circuits

This exercise concerns timing considerations for the computer of Figure 6, which is very similar to the one presented in Lab 7, Exercise D.

Figure 5: Schematic for part of a hypothetical computer design, with an instruction set that is not RISC-V.



Read This First

For a complex combinational component such as an ALU, t_{pd} can be defined as the longest possible delay between the last change on an input wire and the moment when all output wires have correct values. (In fact, I've already made that definition and illustrated it for the example of an adder in Figure 2 on page 3.)

Note that a multiplexer is a special case in which it is not necessary for all input wires to have stable signals before the output is ready. See Figure 7 for an explanation.

For complex sequential components such as the PC, register file, and data memory in the computer of 6 (and also for the computer designs of Chapter 7 of the course textbook) there will be setup time parameters.

For example, if $t_{\text{RFsetup}} = 30\,\text{ps}$ is specified for the register file, that means that a write to a GPR is reliable as long as these three things are true for at least 30 ps in advance of an active clock edge:

- RegWrite = 1;
- the 4-bit A3 input to the Register File (which selects the destination GPR) is stable;
- the 16-bit WD3 input to the Register File (which provides the bit pattern to be copied into the destination GPR) is stable.

To give another example, if $t_{\text{pcq}} = 20\,\text{ps}$ is specified for the PC, that means that the address input to the instruction memory is ready no later than 20 ps after an active clock edge.

What to Do, Part I

Use the schematic of Figure 6 and "Timing Parameters Set 1" from Figure 8 to determine overall t_{pd} values for the following four paths:

- (1) from the PC back to the PC through several elements, including "All Bits 0?";
- (2) from the PC back to the PC through several elements, including the adder that produces the branch target address;

Figure 6: A single-cycle implementation of the Exam16 ISA from Lab 7, Exercise D. This circuit is almost identical to the solution to parts a, b and c of the Lab 7 exercise. Some components have been rearranged to help with the layout of the schematic, and the MemRead output of the control unit has been eliminated.

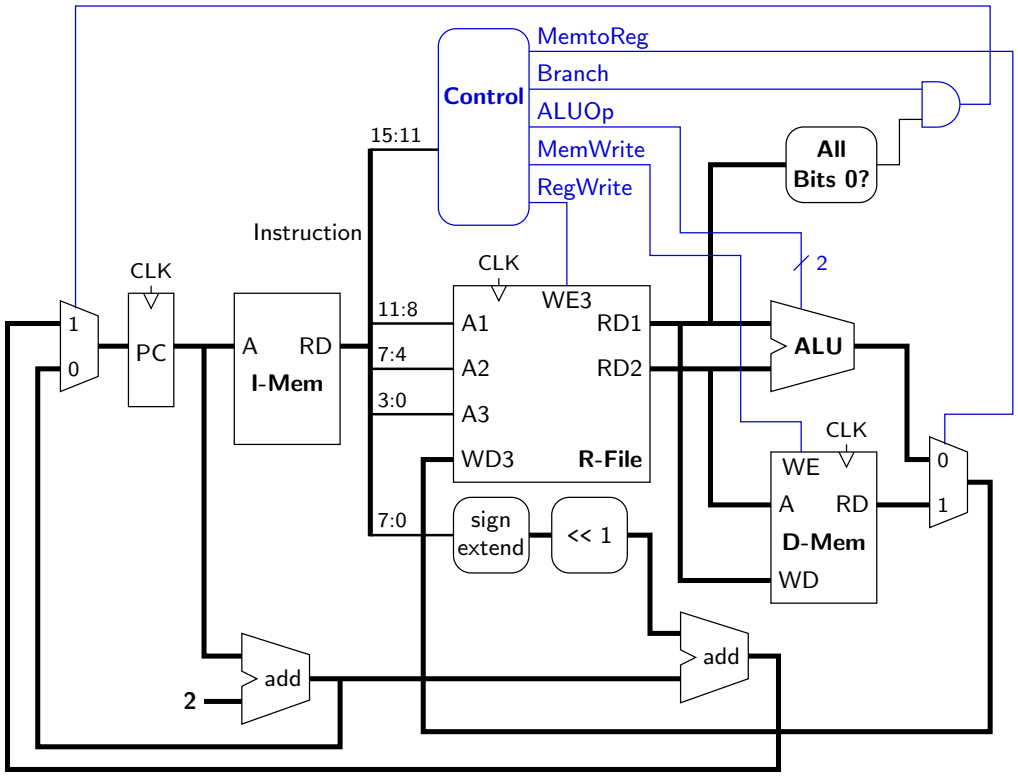


Figure 7: Model for propagation delay in a multiplexer. t_{pd} is the longest possible delay from a change in the select signal (S in this example) or the *selected* data input (D_1 in this example), whichever happens later, to the last change in the output signal. *Unselected* data inputs (D_0 in this example) do not affect the output.

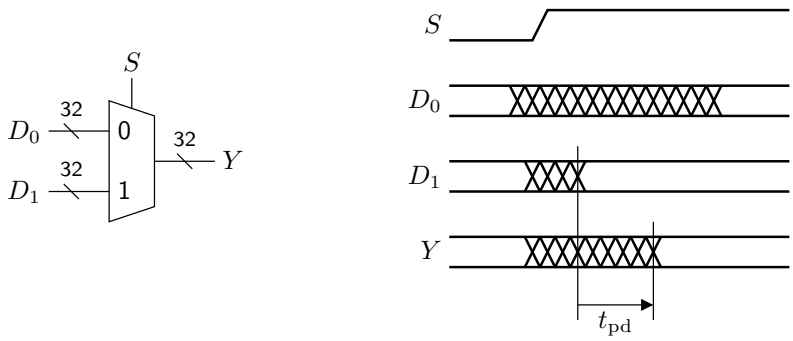


Figure 8: Timing parameters for elements in the computer of Exercise B. All values are in picoseconds.

Timing Parameters Set 1		
element	parameter	value
PC	t_{pcq}	32
	t_{setup}	21
I-Mem	t_{pd}	225
Control	t_{pd}	90
R-File	t_{pd} for read	144
	t_{setup} for write	26
sign extend	t_{pd}	56
<< 1	t_{pd}	40
adder	t_{pd}	170
All Bits 0?	t_{pd}	127
ALU	t_{pd}	194
D-Mem	t_{pd} for read	237
	t_{setup} for write	30
AND gate	t_{pd}	24
2:1 mux	t_{pd}	33

Timing Parameters Set 2		
element	parameter	value
PC	t_{pcq}	32
	t_{setup}	21
I-Mem	t_{pd}	198
Control	t_{pd}	90
R-File	t_{pd} for read	144
	t_{setup} for write	26
sign extend	t_{pd}	56
<< 1	t_{pd}	40
adder	t_{pd}	192
All Bits 0?	t_{pd}	127
ALU	t_{pd}	233
D-Mem	t_{pd} for read	210
	t_{setup} for write	30
AND gate	t_{pd}	24
2:1 mux	t_{pd}	33

- (3) from the PC to the R-File WD3 input through several elements, including the ALU;
- (4) from the PC to the R-File WD3 input through several elements, including D-Mem.

Then determine the minimum clock period to allow reliable operation of the computer, using the t_{pd} values you just calculated, along with t_{pcq} and t_{setup} values from Figure 8.

What to Do, Part II

Repeat Part I using "Timing Parameters Set 2" from Figure 8.

What to Include in Your PDF Submission

Include easy-to-read calculations of t_{pd} values and minimum clock periods.

Exercise C: Tracing instructions though a pipelined processor

Read This First

To learn how a pipelined processor works, it is helpful to trace bit patterns as they move through the pipeline registers.

What to Do

Suppose the processor of textbook Figure 7.51 is implemented with a 1 GHz clock, so its clock period is 1.0 ns. Suppose further that at $t = 72.0$ ns after a program starts running, a positive clock edge occurs that starts the Fetch phase of the `lw` instruction in the following program fragment:

instruction address	*instruction*	*disassembly*
0x0040_0138	0x0000_0013	nop
0x0040_013c	0x0000_0013	nop
0x0040_0140	0x0000_0013	nop
0x0040_0144	0x0000_0013	nop
0x0040_0148	0x0188_2883	lw x17, 24(x16)
0x0040_014c	0x4139_0a33	sub x20, x18, x19
0x0040_0150	0x0000_0013	nop
0x0040_0154	0x0000_0013	nop

The term `nop` is short for "no operation". The machine code for `nop` indicates an R-type instruction that will attempt to write to `x0`, which is guaranteed to have no effect.

The above sequence of instructions is unlikely to appear in a real program, but all the `nop` instructions make this exercise less messy than it would be without the `nop`s.

Assume the following is true at $t = 72.0$ ns:

`x16 = 0x1001_0300, x17 = 0x0000_0246, x18 = 0x0000_0508,`
`x19 = 0x0000_000c, x20 = 0x0000_0bcd.`

The word at data memory address `0x1001_0318` is `6677_8899`.

Answer questions 2–5 below. Question 1 is answered as a model for how to answer all the other questions. Be sure to give valid *reasons* for your answers—please don't just write out a bunch of numbers. Use hexadecimal notation for 32-bit numbers and base two for 5-bit numbers.

1. What gets written into the PC at $t = 73.0$ ns? And what will be the value of InstrD very shortly after $t = 73.0$ ns?

   Answer: That is the end of the Fetch stage of the `lw` instruction, and the beginning of the Decode stage. The PC gets the output of the adder in the Fetch stage, which will be `0x0040_014c`. InstrD gets the instruction, which is `0x0188_2883`.

2. What gets written into the PC at $t = 74.0$ ns? And what will be the value of InstrD very shortly after $t = 74.0$ ns?

   Shortly before $t = 74.0$ ns, what are the values of the following three signals, which are about to be written into the D/E pipeline register: RD1, RdD, and ImmExtD?

3. Shortly before $t = 75.0$ ns, what are the values of the following four signals, which are about to be written into the D/E pipeline register: RD1, RD2, and RdD?

   Also shortly before $t = 75.0$ ns, what is the value of the the 32-bit ALU output?

4. Shortly before $t = 76.0$ ns, what are the values of the following signals: the 32-bit ALU output and RdE?

5. Shortly before $t = 76.0$ ns, what are the values of the following signals: ALUOutM and RdM?

What to Include in Your PDF Submission

Include answers to questions 2–5, with clear reasons for each answer.

Exercise D: Forwarding

What to Do

Using a drawing similar to Figure 7.54 on page 446 of the textbook, show the forwarding paths needed to execute the following sequence of six instructions:

```
and     t0, t1, t2
add     s0, s1, t0
sw      t0, 12(sp)
lw      a0, 0(s0)
sub     t6, s3, s4
add     a1, a0, t6
```

You can make the style of your drawing much simpler than what is in the textbook! There is no need to draw fancy little ALUs and pipeline registers—just draw boxes for pipeline stages and then show all the forwarding.

What to Include in Your PDF Submission

Include your drawing.