Midterm Exam

ENGR3410: CompArch

This midterm is to be completed entirely on your own with no assistance from or collaboration with other humans (NINJAs and professors don't count as human).

You are free to use course slides, your notes, textbooks, online references, Icarus Verilog, etc. *You must document (in writing, including specific titles/URLs) any resources you use other than the course slides*.

You may consult with me or the NINJAs, in person or via private messages on Canvas. Please start early by at least reading through the assignment and asking questions.

I expect this midterm to take less than a week to complete (equivalent of a homework).

Submit by uploading a **PDF** named *first_last_midterm.pdf* and containing your Specification Document, Block Diagram, and Schematics (including sizes) to Canvas. Your work may be typeset or *neatly* handwritten and cleanly scanned.

Please clearly note on your submission the approximate amount of time you spent. This has no impact on your grade and is purely for my calibration purposes.

Problem Summary

Your job for this Midterm is to reproduce part of a real product currently available for sale as faithfully and as cheaply as possible. You will be cloning the LED controller in a bike light.

This bike light has four modes: Off, On, Blinking, and Dim (On at approximately 75% brightness). It cycles through the modes by pressing a single button.

Specification Document [15%]

Write a brief but informative specification document that clearly captures the design intent of the digital electronics portion of the product. This section should include:

- Inputs and Outputs. These are a single button and a single LED, but you need to state that clearly in your document.
- All operational modes of the system
    - Consider showing blink patterns graphically
- Measurements of relevant dimensions in appropriate units with actual numbers
    - E.g. 30Hz, rather than "quickly"

This section should NOT include:

- Information about the mechanical aspect of the product. We're designing the electronics only.
- Any implementation details. Explain **what** the system does, not how it does it.

This should be roughly one page, including figures / charts / FSM diagrams. Another engineer should be able to take your spec document and recreate the device with no additional information. You may wish to include some pseudocode to describe your system if that's helpful to you.

I like http://wavedrom.com/editor.html and http://madebyevan.com/fsm/ for simple figures

Block Diagram [35%]

This is where you begin to answer "**How**". Create a high-level block diagram view of a digital circuit that implements the system laid out in your specification document. Be sure to read the entire assignment before doing this aspect, as there are hints embedded in the scaffolding for the other deliverables. This document is written top-down for context, but can be done in the order that makes you happiest.

Schematics [50%]

Each of the components in your Block Diagram needs to be expanded hierarchically. At the bottom of the hierarchy are basic components: NAND, NOR, AND, OR, XOR, XNOR, NOT, Buffer, D Flip Flop, Multiplexer, Decoder. If you require additional components, they are to be built hierarchically.

For each new component you use, provide the following:

1) Specification. This is a 1-3 sentence description of what the component does.
2) Inputs
3) Outputs
4) Schematic
5) Size of the component in terms transistors (see next section)

Example schematics are given for “pre-baked” components on the following pages; your schematics for other components you create should look similar.

Cost Estimation Model

Estimate the cost of your design in terms of silicon area consumed. For this Midterm, we will assume that area cost is equivalent to transistor count. Non-inverting gates cost 2 extra transistors for the implicit inverter (NOT gate). Express your area costs in transistors.

Gate	**Cost**
NOT	2
2-input NAND/NOR	4
2-input XOR	12
N-input NAND/NOR	2N (N <= 4, for N > 4, build a compound gate)
Edge Triggered D-Flip Flop	26 (see schematic below)

“Pre-Baked” Components

These components are included as scaffolding. You may use them as black boxes without needing to copy them into your own schematic bundle. They are for your reference, and **your design need not use all of them**.

System Clock

Specification

The oscillator sub-circuit provides a 32,768 Hz square wave whenever power is applied. This is used to drive all clocked elements in the system. Do not gate the clock in your design.

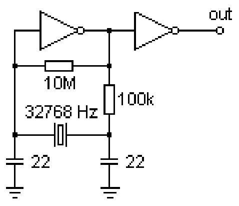
Inputs

No inputs are required for this component; it simply begins oscillating whenever power is applied.

Outputs

clk is a 32,768Hz square wave to drive the clocked logic in the rest of the system.

Schematic



Cost

This uses 2 inverters, and the second drive inverter is slightly larger than usual to drive the clock signal across your chip. You do NOT need to account for this type of electrical sizing for the circuits you design, simply use the sizing model on the previous page. The resistors, capacitors, and quartz crystal are all off-chip and therefore not considered in this calculation.

Subcomponent	Cost per	# Used	Total
Inverter	2	1	2
Drive inverter	8	1	8
			10

Positive Edge Triggered D-Flip Flop

Specification

D	Q	Clk	Q_{next}
0	X	Rising	0
1	X	Rising	1

Inputs

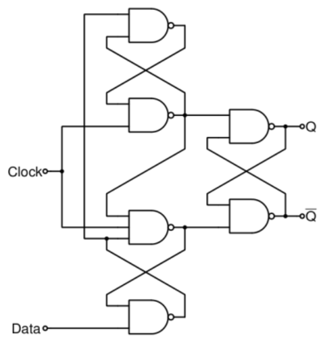
D is the data input. **clk** is the clock input.

Outputs

Q propagates the value of **D** on the positive edge of **clk**

~Q is the Boolean complement of **Q** and is also available as an output.

Schematic



Cost

Subcomponent	Cost per	# Used	Total
2NAND	4	5	20
3NAND	6	1	6
			26

Positive Edge Triggered D-Flip Flop with Enable

Specification

D	E	Q	Clk	Q_{next}
0	0	X	Rising	Q
0	1	X	Rising	0
1	0	X	Rising	Q
1	1	X	Rising	1

Inputs

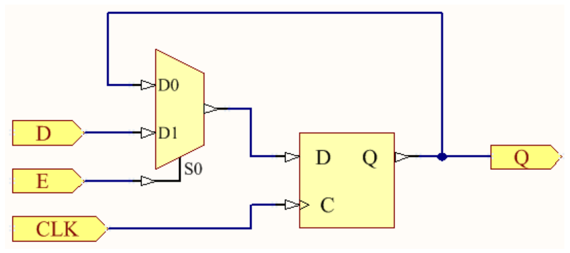
D is the data input. **clk** is the clock input. **E** is the enable input

Outputs

Q propagates the value of **D** on the positive edge of **clk** if and only if **E** is true

~Q is the Boolean complement of **Q** and is also available as an output.

Schematic



Cost

Subcomponent	Cost per	# Used	Total
DFF	26	1	26
2-Input MUX	14	1	14
			40

LED Driver

Specification

The LED Driver amplifies a logic signal to a level that can electrically power the LED.

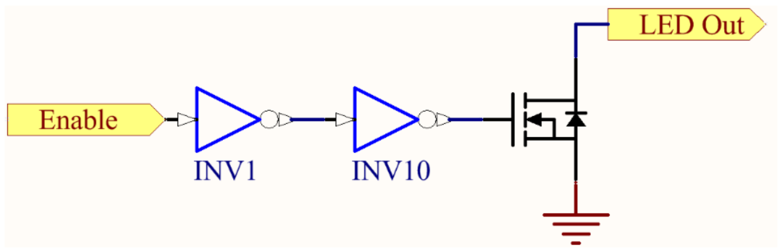
Inputs:

> **Enable** – Active High Input. When this input is high, the LED is provided power and will turn on.

Outputs:

> **LED Out** – Open Drain Output. Sinks current to ground when active. High impedence (disconnected) when not active. Connect to the cathode of the LED. Connect anode of LED to positive power rail through limiting resistor to ensure no more than 20mA.

Schematic



Cost

The final output transistor is oversized in order to handle the LED current. It consumes 150 transistors worth of die area. The inverter that drives this transistor is 10 times larger than normal. The first inverter is normal sized, and is 2 transistors. The total cost is therefore 172.

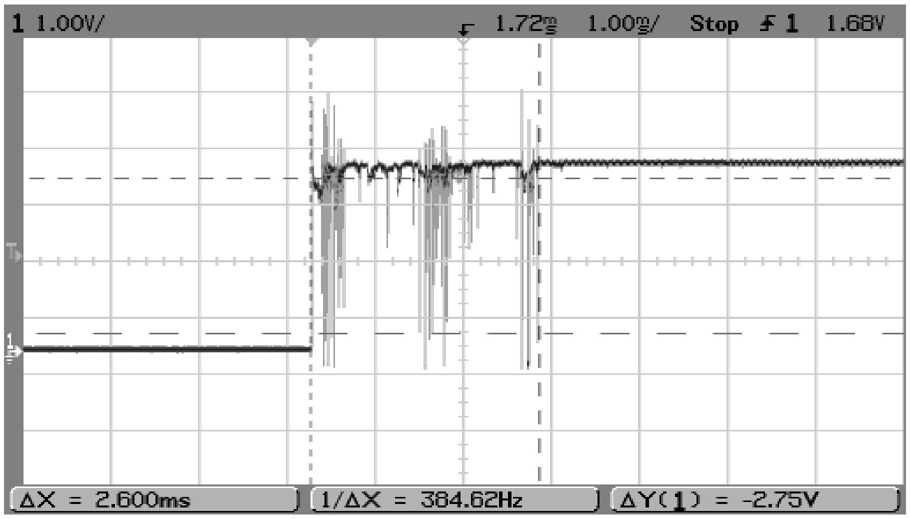
Subcomponent	Cost per	# Used	Total
INV1	2	1	2
INV10	20	1	20
Drive transistor	150	1	150
			172

Interfacing with the Outside World

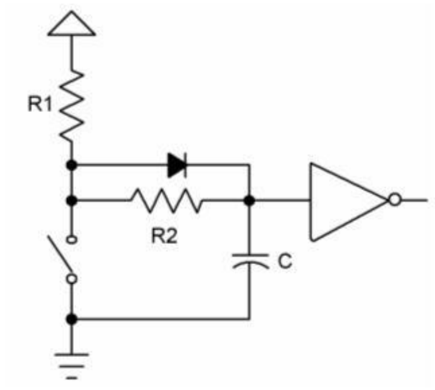
The input to this system is a physical button pushed by a human finger.

Debouncing

Because of their inherent mechanical nature, buttons and switches do not make clean transitions between logic values. Instead, they may noisily oscillate many times for each button press, a condition known as "bouncing" or "ringing".



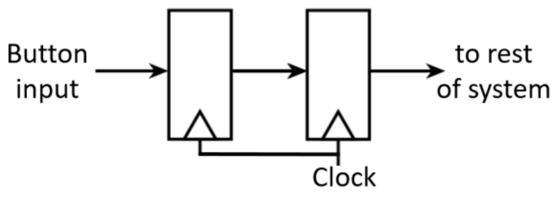
Often a "debouncer" component follows the synchronizer and filters out button noise, guaranteeing that each button press or release only results in a single clean transition as observed by the rest of your system. Here is a simple RC debouncer circuit, which you can read all about here: http://www.ganssle.com/debouncing-pt2.htm



You may assume that the total die area for the debounce circuit roughly equivalent to 200 transistors for the purpose of calculating your total area cost.

Synchronization

In our digital circuits, we require that signals transition relative to the clock such that setup and hold times are met for flip-flops. Since the user can push the button at any time, there is no guarantee that this requirement will be met. To deal with this, we can use a "synchronizer" component.



Include two D-Flip Flops in series at the output of your debouncer to synchronize the button press so that transitions are aligned with your clock domain[1].

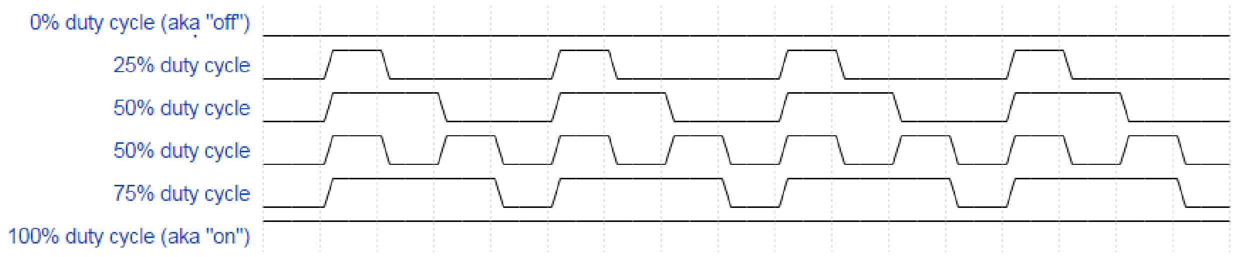
Partial Components / Hints

Counters

Since the system oscillator is several orders of magnitude faster than your blink patterns, you could use some form of counter as part of the subsystem that generates blink timing, as we did in Homework 3. One way to think of a counter is as a specialized form of finite state machine. As such, you can choose how the count state is encoded – one hot (requires N flip-flops) or binary ($\log_2 N$ flip-flops), each with their own advantages.

Dimming

Digital circuits like this one often implement analog behaviors such as dimming using Pulse Width Modulation. This technique turns the LED off and on very rapidly, faster than the eye can see. The perceived brightness depends on the total amount of time the LED spends on, called the "duty cycle".



Tools

This Midterm does not require the use of Icarus Verilog or the Xilinx tools – there is no Verilog anywhere on this! You may use them if you'd like to try out your ideas; they have tons of helpful features.

[1] Synchronizers cannot guarantee that the signal will be resolved in bounded time due to the potential for metastability, but in practice we can reduce the probability of failure to an arbitrarily low acceptable rate.

Challenge problems

These are completely optional and just for fun if you have extra time.

Challenge 1

Optimize your design for size. Smallest (correct) controller design wins bragging points

Challenge 2

LED controllers sometimes contain a “breathe” function: Think of a Mac in sleep mode. The LED starts off, its duty cycle is slowly increased to a peak value, and then slowly decreased back to zero. For extra credit, design a digital circuit that will show this behavior. Assume that the human vision flicker limitation is 128 Hz; keep your flicker rate faster than this.

Target a total cycle period of 4 seconds. Linear duty cycle ramping is easiest.

Challenge 3

I also have a rear bike light, with 3 LEDs and more complex blink patterns. Design an FSM and controller for this light.