CompArch: Midterm

This midterm is to be completed entirely on your own with no assistance from or collaboration with other humans.

You are free to use course slides, your notes, textbooks, online references, ModelSim, 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 questions on Piazza. Please start early - we will stop offering assistance other than simple clarification questions starting 48 hours before the deadline.

Due date: before midnight on Sunday, November 15

Submit by pushing your Specification Document, Block Diagram, and Schematics (with sizes) to GitHub as PDF or MarkDown.

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 50% brightness). It cycles through the modes by pressing a single button.

Specification Document

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. 10Hz, rather than "quickly"

This section should NOT include:

- Any information about the mechanical aspect of the product
- Any implementation details. Explain **what** it 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.

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

Block Diagram

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.

Schematic

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 of the number of Gate Inputs it uses.

Cost Estimation Model

Estimate the cost of your design in terms of silicon area consumed. For this Midterm, we will assume that cost linearly scales with the number of gate inputs for basic inverting gates. Non-inverting gates cost 1 extra for the implicit inverter. Express your area costs in Gate Input Equivalents (GIE).

Gate	**Cost**
Inverter	1
2 Input NAND gate	2
2 Input AND gate	2+1=3
N Input NOR gate	N
Edge Triggered D-Flip Flop	13

“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.**

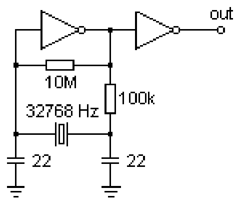
Inputs

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 for a total of 2 GIE worth of space. The resistors, capacitors, and quartz crystal are all off-chip and therefore not considered in this calculation.

Subcomponent	Cost per	# Used	Total
Inverter	1	2	2
			2

Positive Edge Triggered D-Flip Flop

Specification

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

Inputs

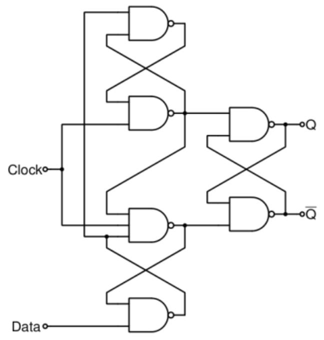
D is the data input. **clk** is the clock input.

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	2	5	10
3NAND	3	1	3
			13

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

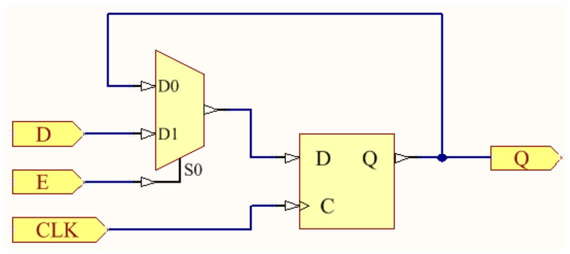
D is the data input. **clk** is the clock input. **E** is the enable input

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	13	1	13
2Mux	7	1	7
			20

N stage Ring Counter

Specification

This parameterized component is a one-hot counter. When **E** is asserted, the hot bit moves one element down the ring each positive **clk** edge.

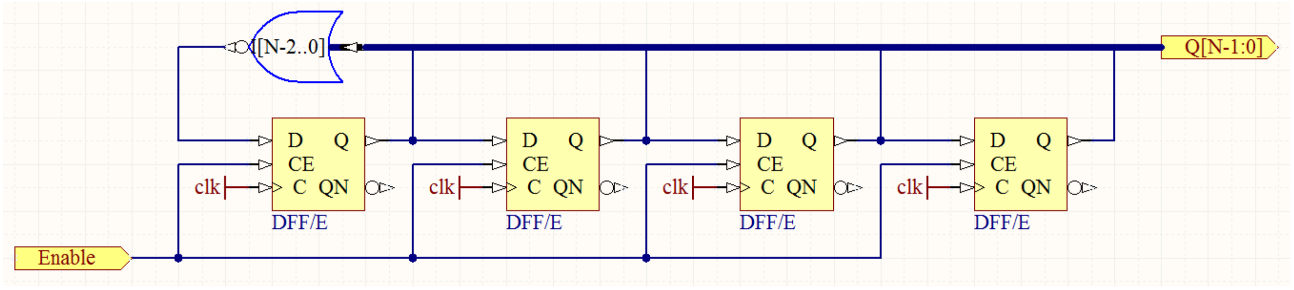
Inputs

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

Outputs

Q[N-1:0] has exactly one bit high, all others are low. This high bit rotates through the group each positive **clk** edge when **E** is asserted high.

Schematic



Note: The NOR gate restarts the cycle when the hot bit is at the end of the ring. This structure removes the need for a reset signal, as the ring will eventually enter a normal operating pattern regardless of the initial flip-flop values (prove this to yourself).

Cost

Subcomponent	Cost per	# Used	Total
DFF w/E	20	N	20N
(N-1)NOR	N-1	1	N-1
			21N-1

LED Driver

Specification

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

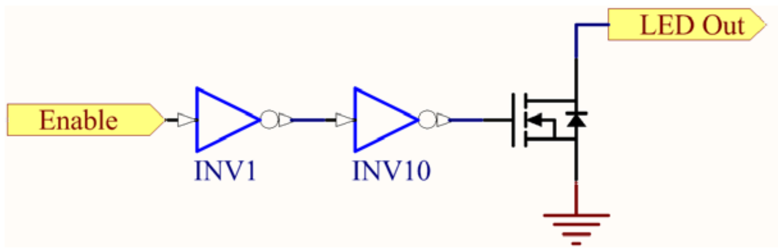
Inputs:

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 the 200 Gate Input Equivalent space. The inverter that drives this transistor is 10 times larger than normal. The first inverter is normal sized, and is 1 Gate Input Equivalent. The total cost is therefore 211 GIE.

Subcomponent	Cost per	# Used	Total
INV1	1	1	1
INV10	10	1	10
Drive transistor	200	1	200
			211

Partial Components / Hints

Up Counter

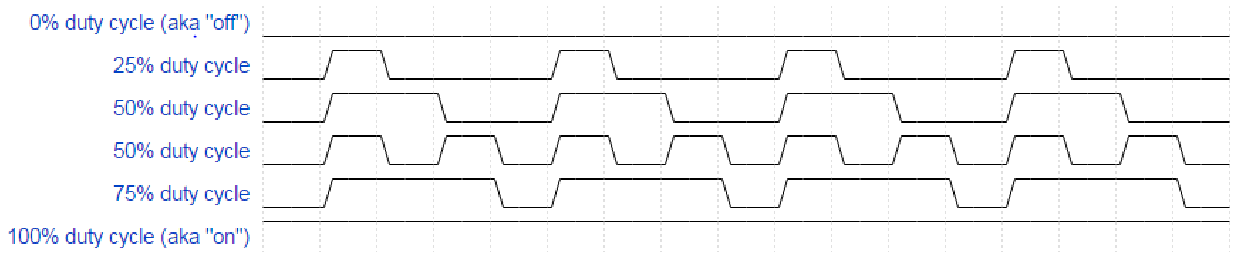
An up-counter is the binary encoded version of the one-hot ring counter. This means that it needs $\log_2 N$ D Flip Flops instead of N. Since the system oscillator is several orders of magnitude faster than your Blink Patterns, I would suggest using an Up Counter as part of the subcircuit that generates that timing.

Input Conditioner

You will need to condition the button input. Assume that the button noise / bouncing decays within 1 millisecond. This will require careful selection of when to use a binary encoding and when to use a one hot encoding.

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 ModelSim or the Xilinx tools – there is no Verilog anywhere on this! You may use them if you'd like, they have tons of helpful features.

Challenge problems

Challenge 1

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

Challenge 2

LED controllers often 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.