ECE M16 Final

Lawrence Liu

August 11, 2022

Problem 1

$$
\begin{array}{r|cccc}
 & 1 & 1 & 0 & 1 \\ \cline{2-5}
1 & 11 & 00 & 00 & 10 \\
 & 1 & & & \\ \cline{2-2}
101 & 10 & 00 & & \\
 & 1 & 01 & & \\ \cline{2-3}
1100 & & 11 & 00 & \\
 & & 0 & & \\ \cline{3-4}
 & & 11 & 00 & 10 \\
11001 & & 1 & 10 & 01 \\ \cline{3-5}
 & & & & \\
 & & 1 & 10 & 01
\end{array}
$$

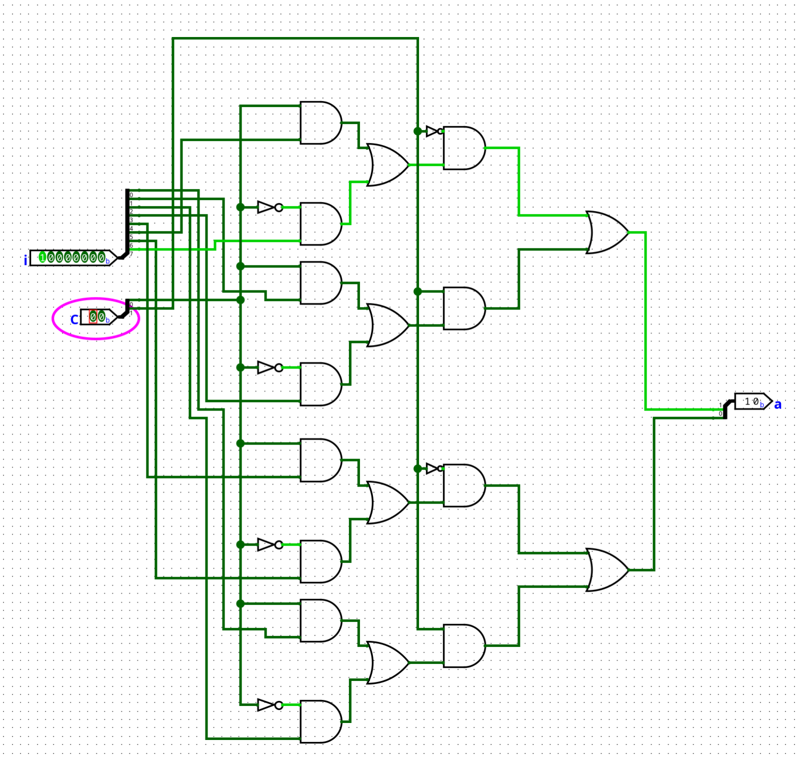
Problem 2

Basing on the assumption that $c[1:0] = 11$ corresponds with $o[1:0] = i[1:0]$ we have that

$$
\begin{aligned}
a[1] &= \overline{c[1]}.\overline{c[0]}.i[7] + \overline{c[1]}.c[0].i[5] + c[1].\overline{c[0]}.i[3] + c[1].c[0].i[1] \\
&= \overline{c[1]}.(\overline{c[0]}.i[7] + c[0].i[5]) + c[1].(\overline{c[0]}.i[3] + c[0].i[1])
\end{aligned}
$$

$$
\begin{aligned}
a[0] &= \overline{c[1]}.\overline{c[0]}.i[6] + \overline{c[1]}.c[0].i[4] + c[1].\overline{c[0]}.i[2] + c[1].c[0].i[0] \\
&= \overline{c[1]}.(\overline{c[0]}.i[6] + c[0].i[4]) + c[1].(\overline{c[0]}.i[2] + c[0].i[0])
\end{aligned}
$$

Which results in a circuit like this



Problem 3

I created the circuit, and it is shown above, and I tested it with the following python checker srcipt

```python
import numpy as np
import pandas as pd
import os
from calendar import monthrange

def RunCircuit(logisim_jar  : str, circuit : str):
    """
    This function runs the logisim simulator and returns the output of the circuit as
    a  pandas dataframe.
    """
    output=os.popen(f"java -jar {logisim_jar} {circuit} -tty table").read()
    output=[o.split() for o in output.split("\n")[:-1]]
    return pd.DataFrame(output[1:],columns=output[0])

def checkQ2(truth_table:pd.DataFrame)->bool:
    """
    This function checks the output of the circuit for the truth table and returns
    weather the output is correct or not.
```

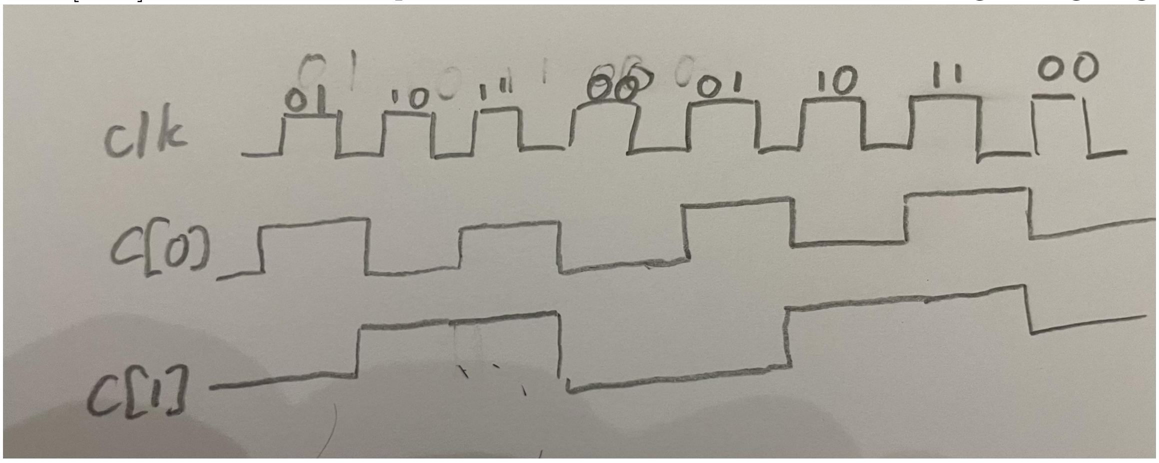
```python
    """
    #convert hex to binary
    truth_table['i']=truth_table['i'].apply(lambda x: f'{int(x,16):0>8b}')
    for i,row in truth_table.iterrows():
        c=int(row.C,2)
        i=row.i
        a=row.a
        #calculate a expected
        a_expected=i[c*2:c*2+2]
        #check if a is equal to a_expected
        if a!=a_expected:
            print("error!")
            print(f"at c={row.C}")
            print(f"expected a={a_expected}")
            print(f"got a={a}")
            print(f"i={row.i}")
            return False
    return True

if __name__=="__main__":
    truth_table=RunCircuit("../logisim-evolution.jar","logisim/FinalQ3.circ")
    if checkQ2(truth_table):
        print("Q2 passed!")
```

This script utilizes Logisim's command line ability. I had the files in the following format

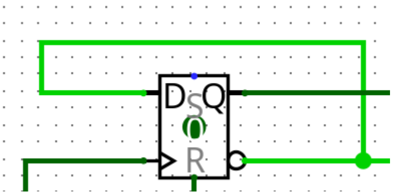
```
ECEM16
 |- .gitignore
 |- Final
 | |-logisim
 | | |- FinalQ3.circ
 | :
 | :
 | |- checker.py
 |- .gitignore
 |- logisim-evolution.jar
```

Problem 4

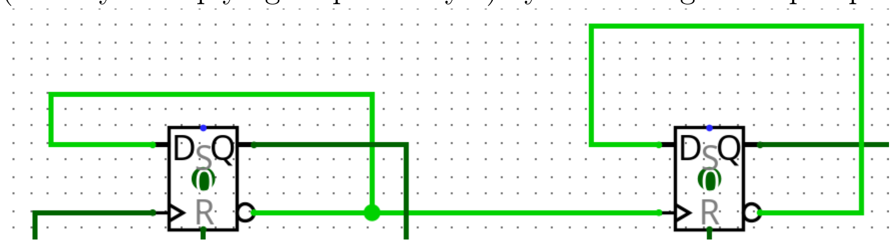
Let $c[1:0]$ be the desired output from our counter: we want the following timing diagram:



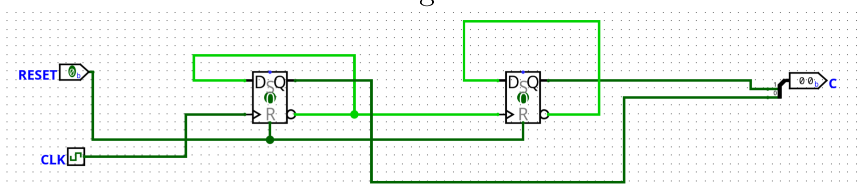
The first thing we notice is that $c[0]$ is just the clock input but with a period twice of clock, and $c[1]$ is just the clock input with a period of 4 times clock. We can make the output from a circuit toggle with each clock pulse (ie doubling its frequency), by connecting $\overline{Q}$ with D, in the following manner:



Likewise we can make the output from a circuit toggle with every other clock pulse (thereby multiplying its period by 4) by connecting two flip flops in the following manner

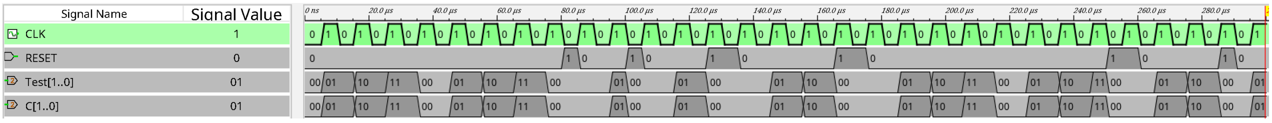


Therefore we have the following circuit:



Problem 5

I created the circuit, and it is shown above, and I tested by comparing its output vs that of a logisim counter, its output is denoted as test in the chronograph below.



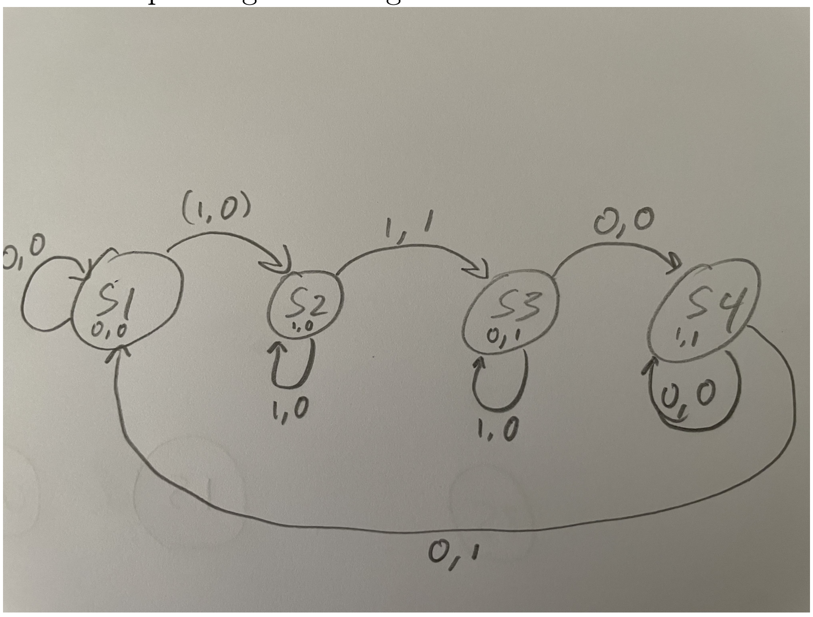
As one can see, the chronograph outputs of my counter vs logisim's counter is the same.

Problem 6

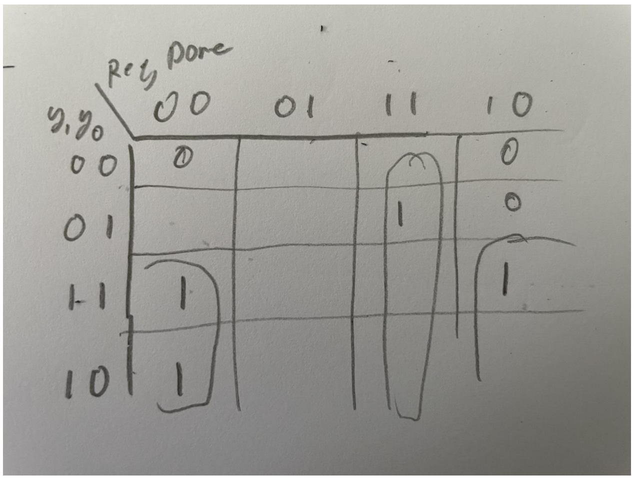
We have the following transition table:

	Prev state	Inputs (REQ,DONE)				OUTPUT	
	$(y_1 y_0)^n$	0,0	0,1	1,1	1,0	GO	ACK
S1	0,0	S1			S2	0	0
S2	0,1			S3	S2	1	0
S3	1,1	S4			S3	0	1
S4	1,0	S4	S1			1	1

The corresponding state diagram is shown below.



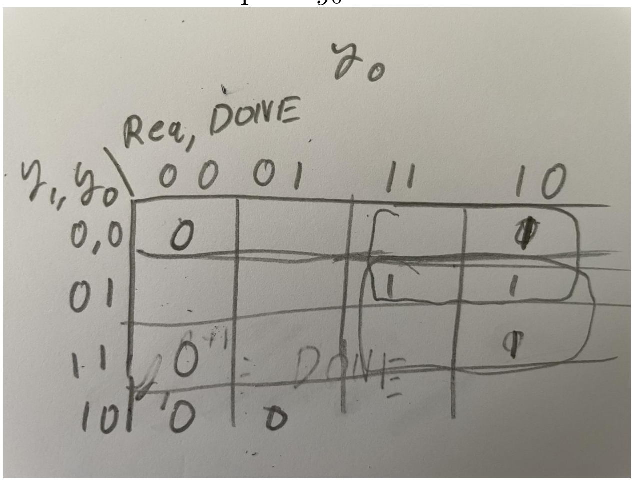
And it results in the following kmap for y_1



Which corresponds with the equation:

$$y_1^{n+1} = y_1^n.\overline{REQ} + DONE.REQ$$

Likewise the kmap for y_0 is shown below.



Which corresponds with the equation:

$$y_0^{n+1} = y_0^n.REQ + \overline{y_1^n}.REQ$$

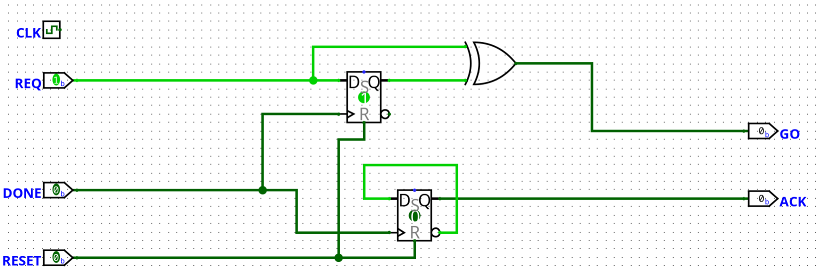
We have that

$$GO^{n+1} = y_1^{n+1}.\overline{y_0^{n+1}} + \overline{y_1^{n+1}}.y_0^{n+1}$$

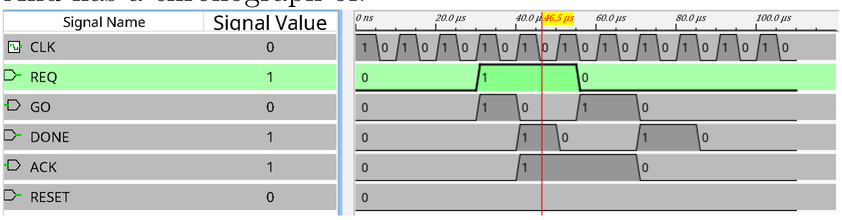
And

$$\begin{aligned} ACK^{n+1} &= y_1^{n+1} \\ &= y_1^n.\overline{REQ} + DONE.REQ \end{aligned}$$

Problem 7

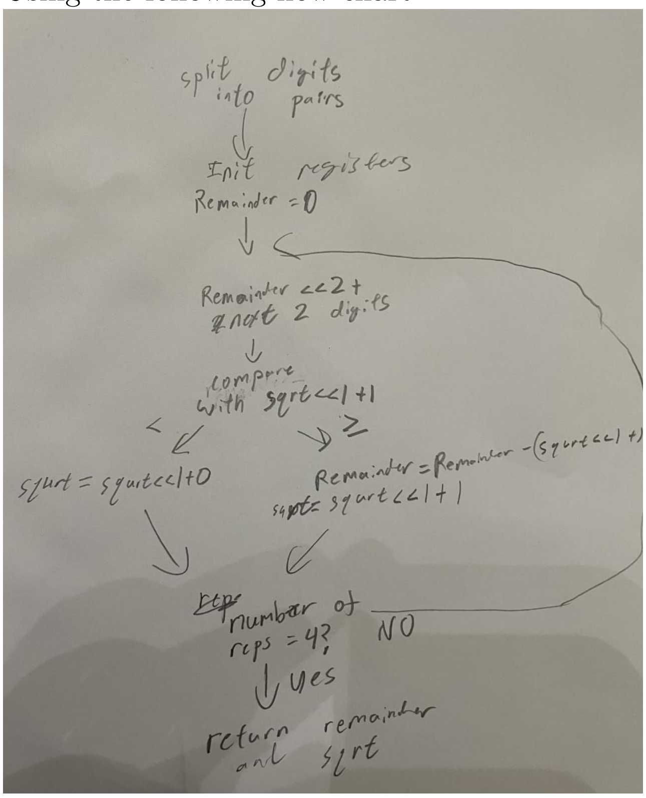


And has a chronograph of:

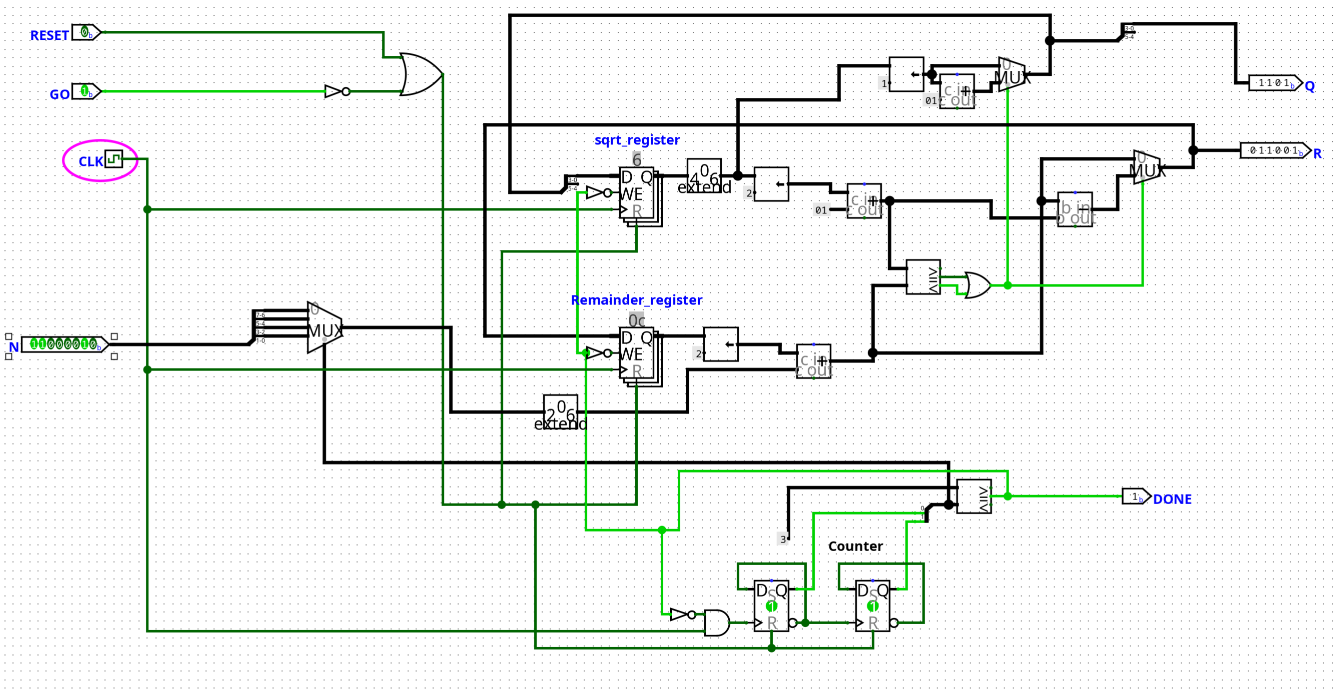


Problem 8

Using the following flow chart



I created the circuit


RESET
GO
CLK
N
sqrt_register
Remainder_register
Counter
Q
R
DONE
MUX
extend