Course code: CE118.P13
Lecturer: Ms. Ho Ngoc Diem

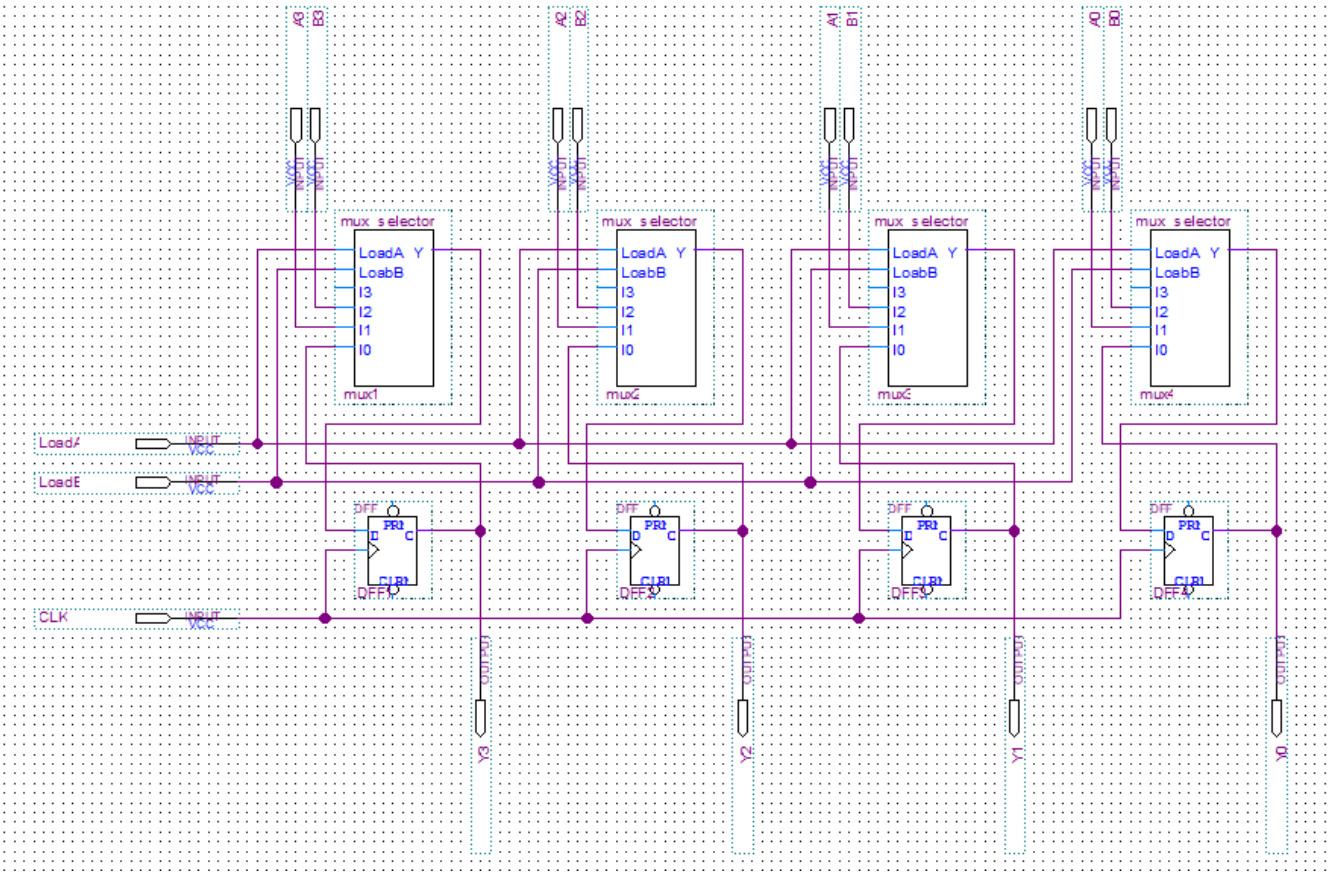
Name: Tran Ngoc Anh
Student code: 22520077

ASSIGNMENT 3

(from Principles of Digital Design Reference Book)

Chapter 7 – Storage Components

7.2 (Registers) Design a register with two load signals that enable the loading of data from two different sources.



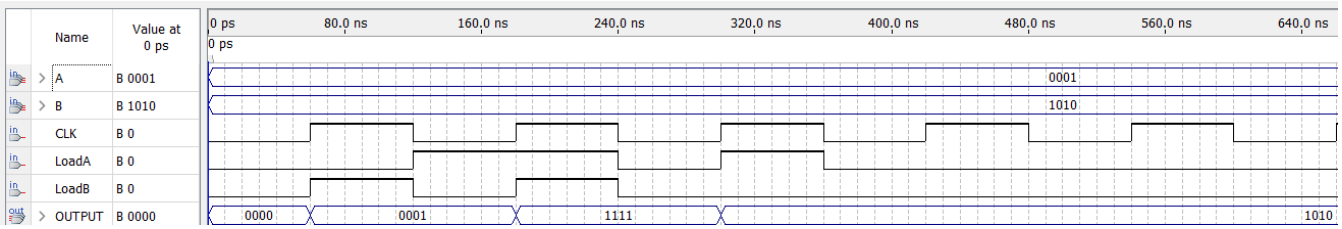
The circuit consists of a 4-bit register with four multiplexers (MUX4-to-1) per bit, controlled by the load signals. The multiplexers allow data to be selected from two sources (A and B), based on the values of LoadA and LoadB. The register is implemented using D flip-flops, which hold the output values (Y0, Y1, Y2, Y3), synchronized by a clock signal (CLK).

The truth table:

LoadA	LoadB	Operation
0	0	Not change
0	1	Load A
1	0	Load B
1	1	undefined

Proved by waveform:

- Set CLK = 60.
- Set A = 0001, B = 1010
- LoadA, LoadB are the values to be checked in the truth table respectively.



7.3 (Registers) Explain the difference between resetting the register and loading an all-0's input into the register.

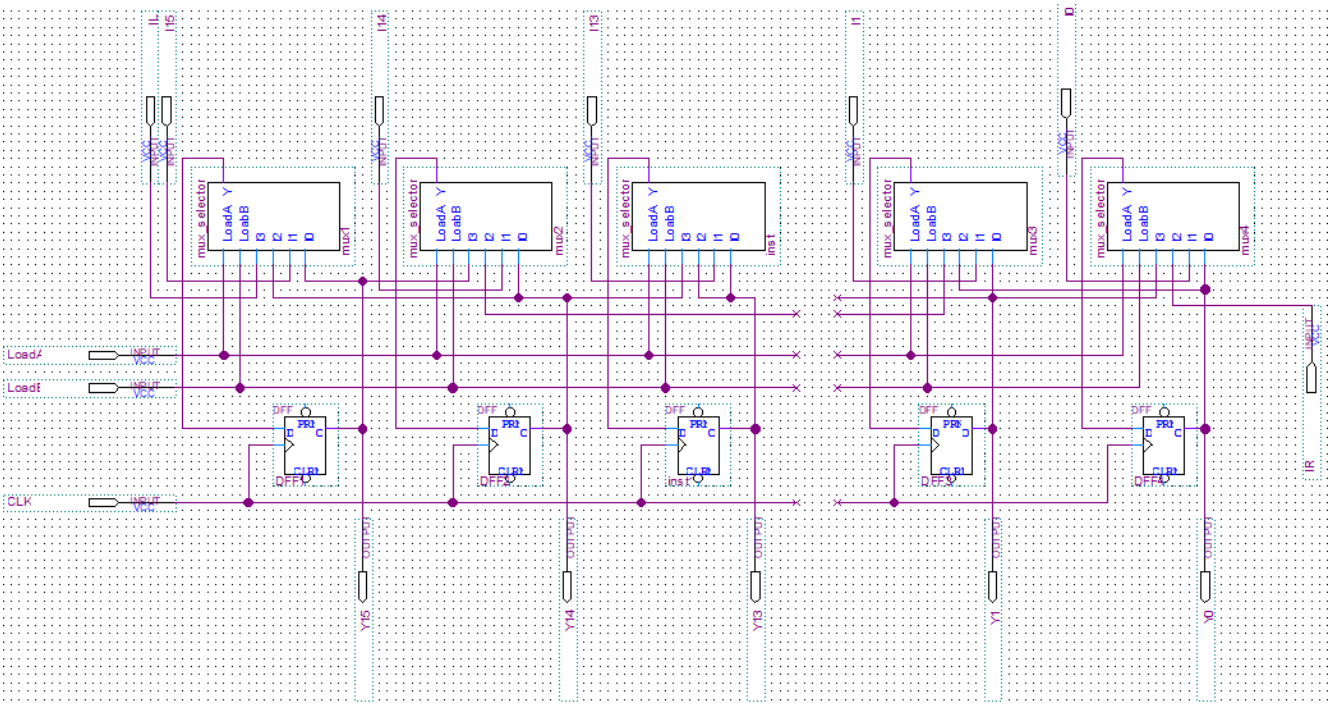
Sự khác biệt nằm ở cơ chế thực hiện:

- Reset: khi tín hiện Reset được kích hoạt, nội dung của thanh ghi bị ghi đè về 0 trực tiếp.
- Nạp (loading) 0 vào thanh ghi: các đầu vào dữ liệu được đặt về 0 và thanh ghi được yêu cầu nạp vào giá trị này. Kết quả là thanh ghi có tất cả nội dung 0 nhưng không có sự ghi đè như Reset.

7.4 (Register) Design a 16-bit register that can load new data and rotate or shift its content left or right.

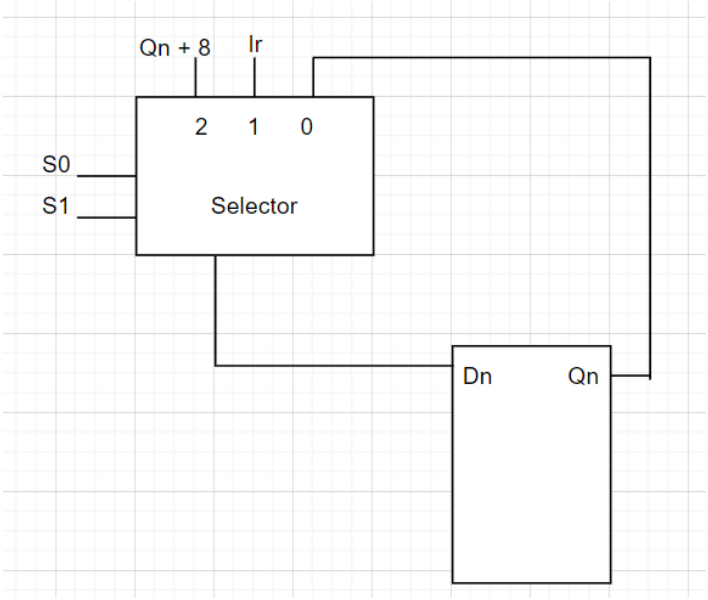
S1	S0	Operation	Q15	Q14 ... Q1	Q0
0	0	No change	Q15	Q14 ... Q1	Q0
0	1	Load	I15	I14 ... I1	I0
1	0	Shift left	Q14	Q13 ... Q0	IR
1	1	Shift right	I2	Q15 ... Q2	Q1

(NOTE*: khoảng x....x là 10 MUX và D flipflop tương tự từ I12 đến I2)

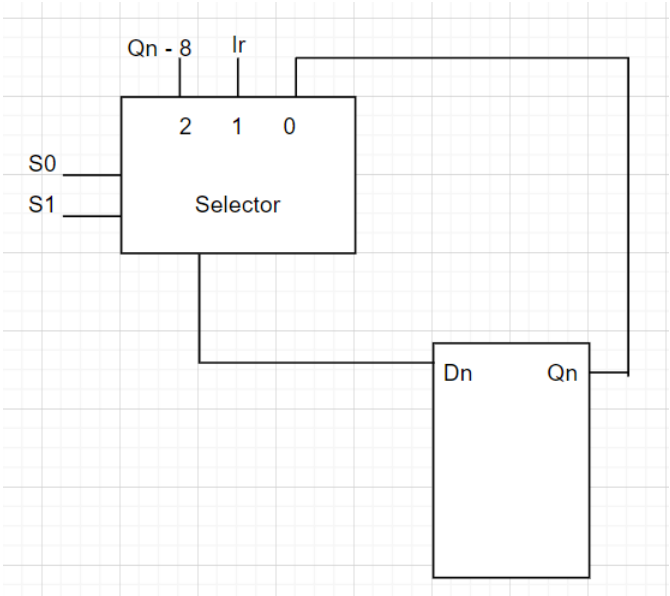


7.6 (Shift registers) Design a 16-bit register that can perform two operation: (1) load a new data and (2) swap the most-significant and least-significant bytes.

From bit-7 to bit-0, we have:



From bit-15 to bit-8, we have:



7.8 (Counters) Design a binary counter that counts up only in:

a) Even numbers (0, 2, 4, 6, 8,...)

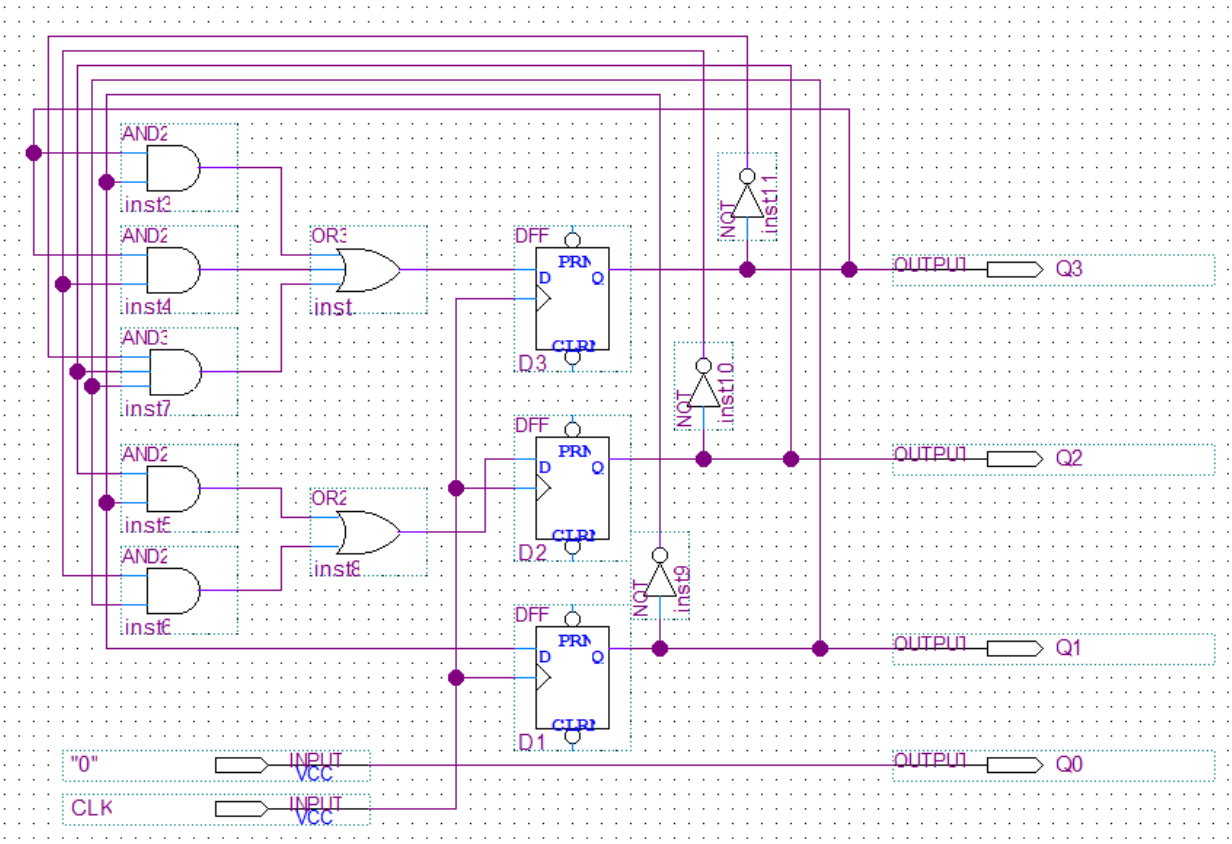
The truth table:

Current state				**Next state**			
Q3	Q2	Q1	Q0	Q3(next)	Q2(next)	Q1(next)	Q0(next)
0	0	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	1	0
0	1	1	0	1	0	0	0
1	0	0	0	1	0	1	0
1	0	1	0	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	0	0	0	0

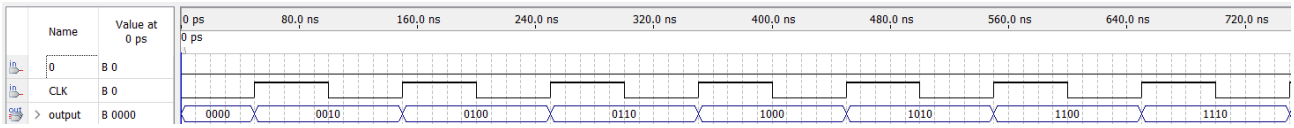
After drawing the Karnaugh cover and simplifying, we get the next-state equations:

- Q3(next) = Q3Q1' + Q3Q2' + Q3'Q2Q1
- Q2(next) = Q2Q1' + Q2'Q1
- Q1(next) = Q1'
- Q0(next) = 0

Logic schematic:



Proved by waveform:



b) Odd numbers (1, 3, 5, 7, 9,...)

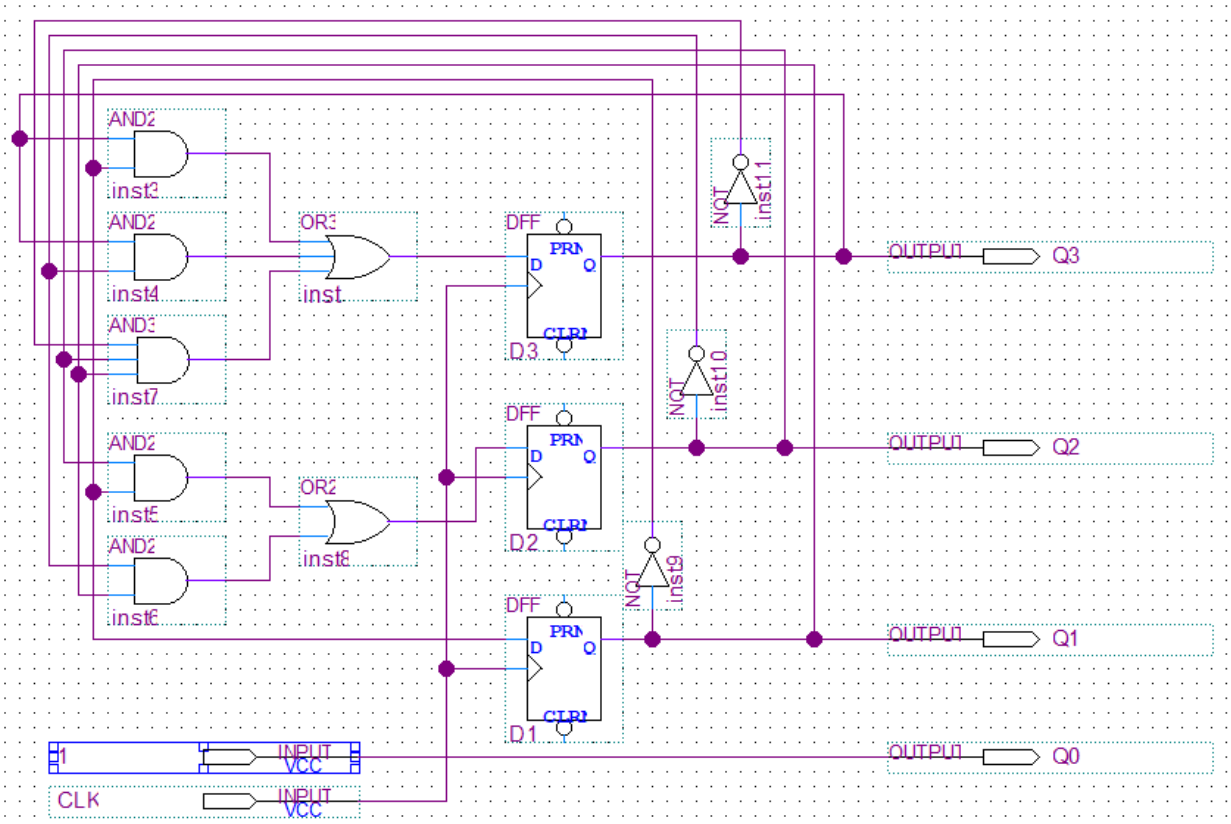
The truth table:

Current state				Next state			
Q3	Q2	Q1	Q0	Q3(next)	Q2(next)	Q1(next)	Q0(next)
0	0	0	1	0	0	1	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	1	1
0	1	1	1	1	0	0	1
1	0	0	1	1	0	1	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	1

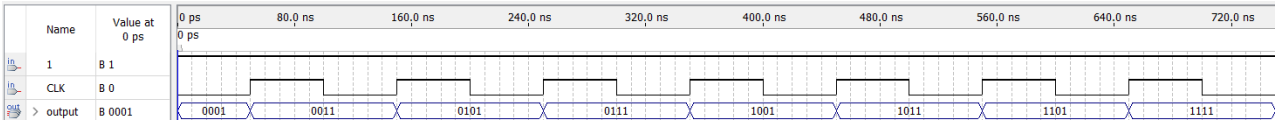
After drawing the Karnaugh cover and simplifying, we get the next-state equations:

- Q3(next) = Q3Q1' + Q3Q2' + Q3'Q2Q1
- Q2(next) = Q2Q1' + Q2'Q1
- Q1(next) = Q1'
- Q0(next) = 1

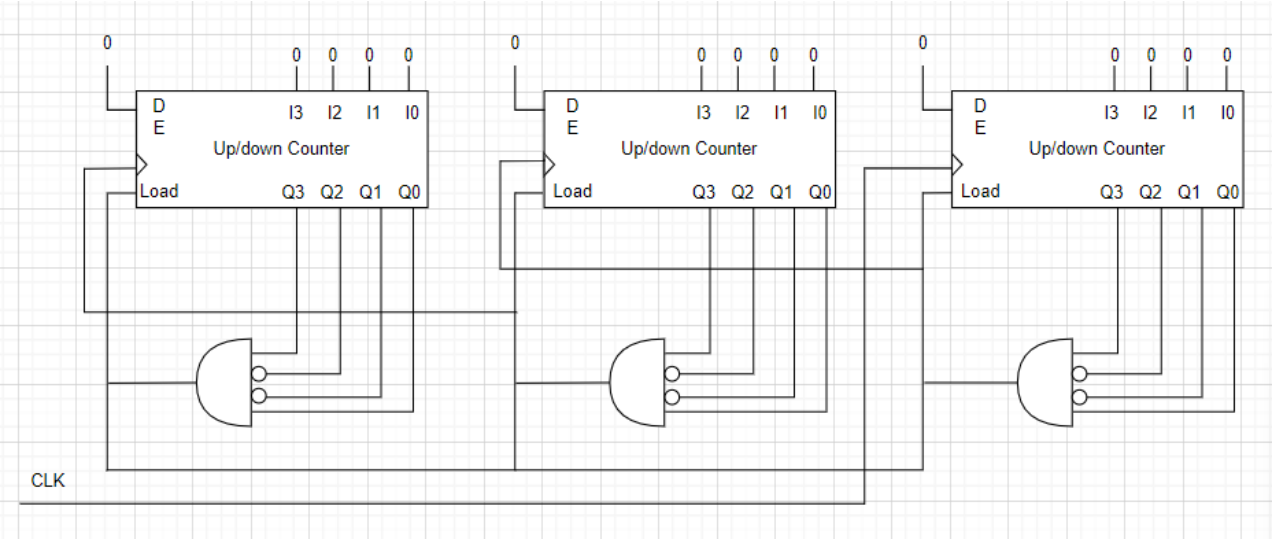
Logic schematic:



Proved by waveform:



7.11 (Counters) Design a decimal counter that counts modulo 1000.



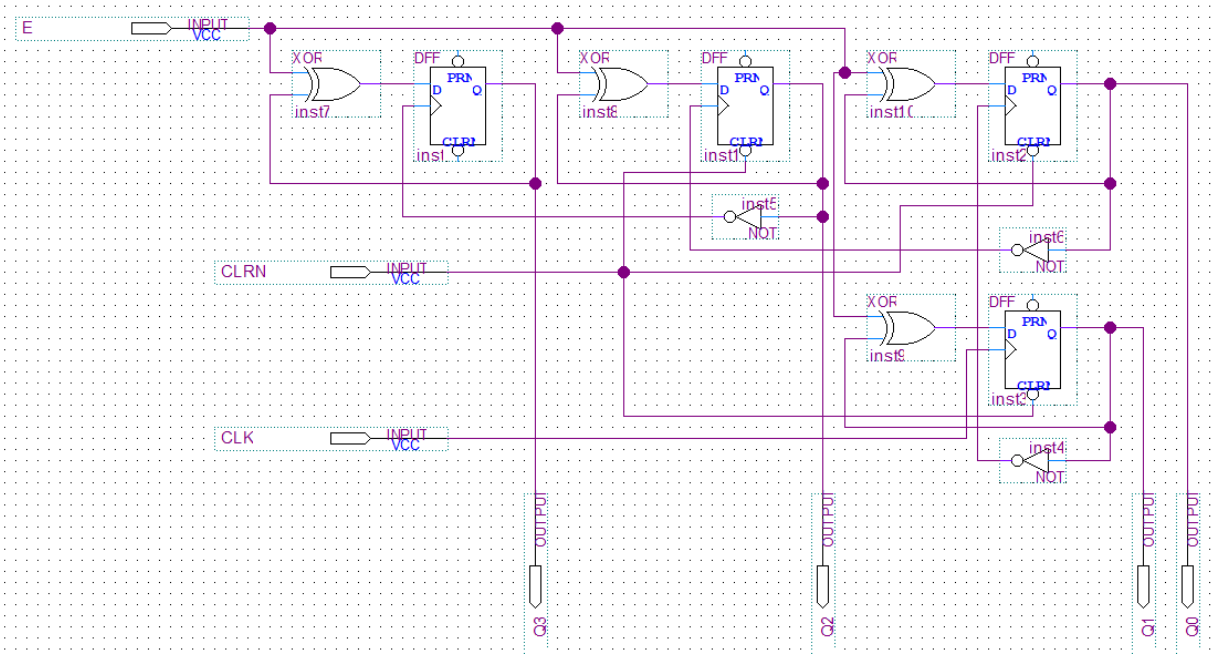
7.13 (Asynchronous counters) Contrust a 4-bit asynchronous counter using:

a) D flip-flops

Since the asynchronous counter uses T flipflop, to design a synchronous counter using D flipflop we can change the circuit based on the truth table.

Qi	E	Di
0	0	0
0	1	1
1	0	1
1	1	0

⇨ D = E xor Q so D flipflop can excute as T flipflop.



b) JK flip-flops

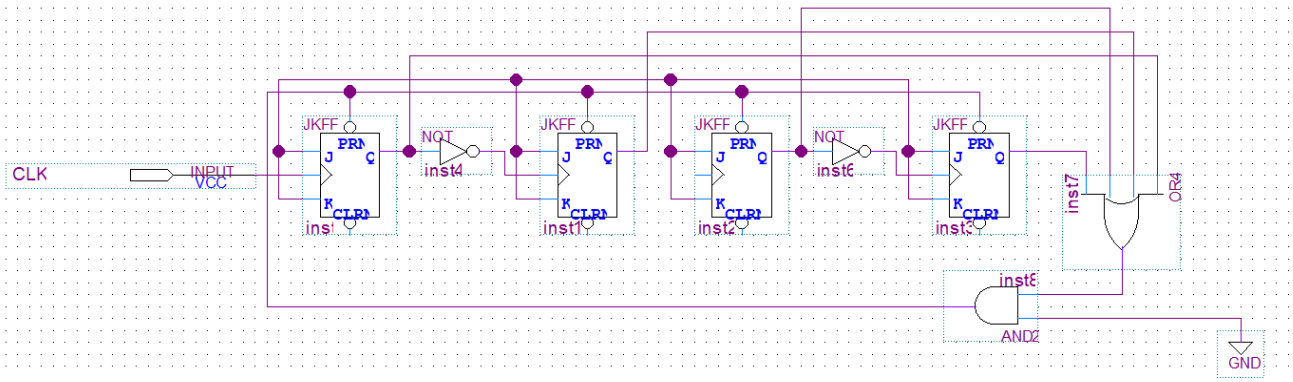
Using 4 flipflops JK0, JK1, JK2, JK3

JK0: J0 = 1, K0 = 1 (kích cạnh lên)

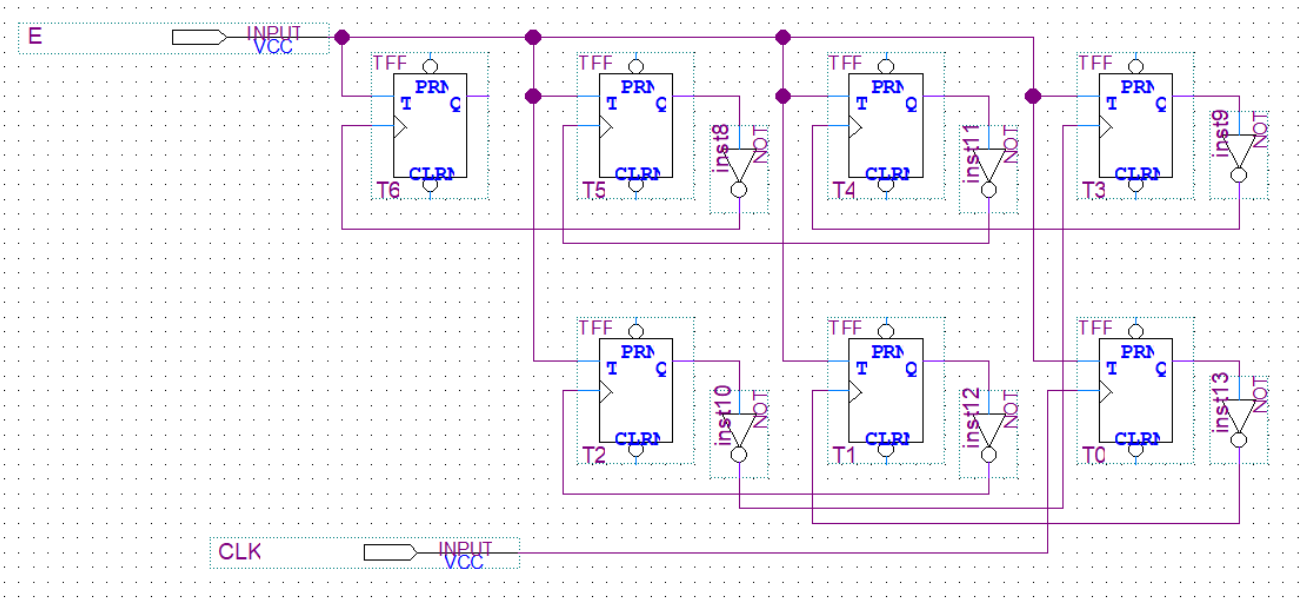
JK1: J1 = Q0, K1 = Q0 (toggle khi Q0 kích cạnh lên)

JK2: J2 = Q1, J2 = Q1 (toggle khi Q1 kích cạnh lên)

JK3: J3 = Q2, J2 = Q1(toggle khi Q2 kích cạnh lên)

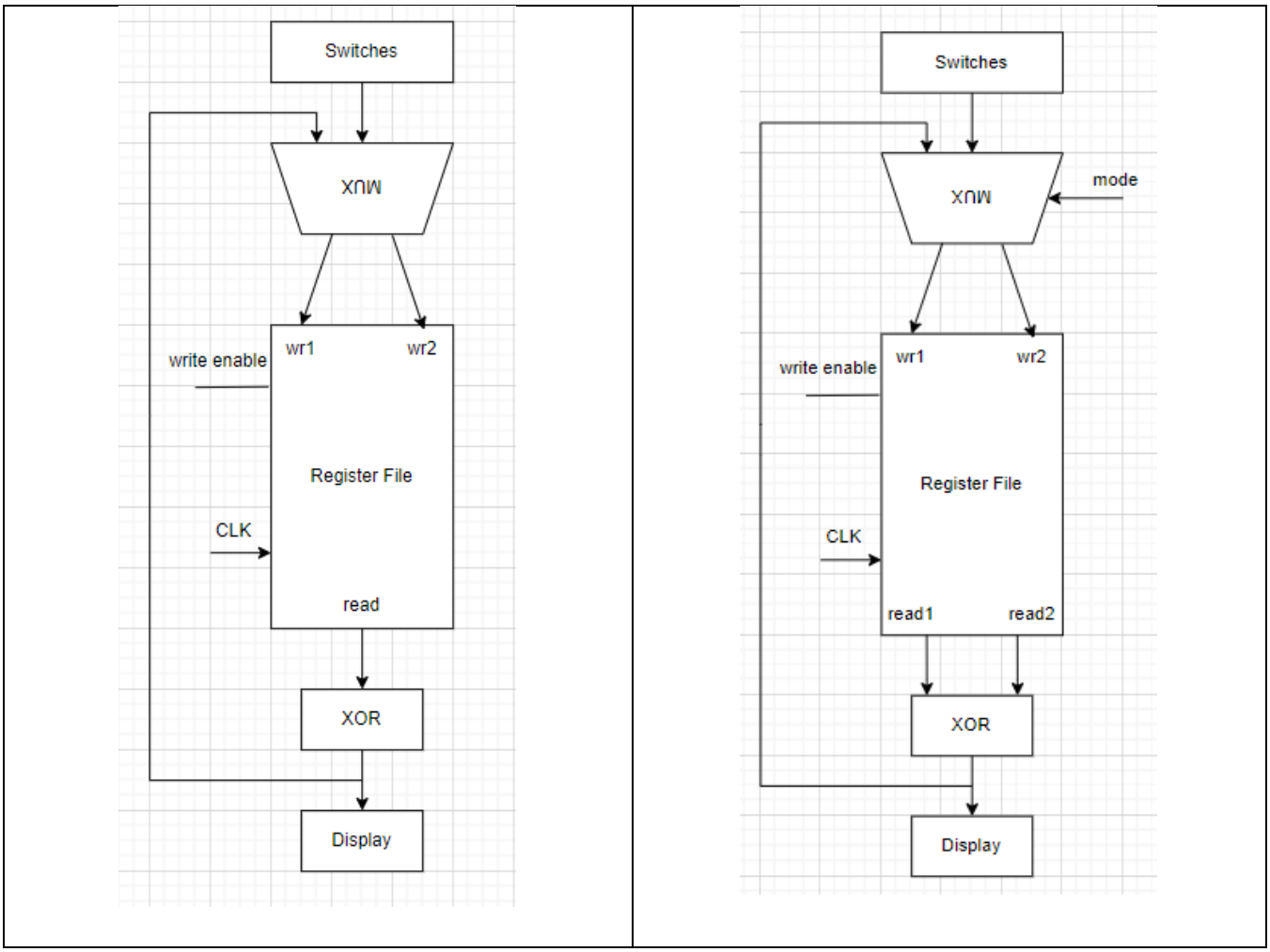


7.14 (Asynchronous counters) Design a decimal modulo-100 asynchronous counter.



7.15 (Register files) Design an 8 x 4 register file with:

b) Two write and one read ports	c) Two write and two read ports



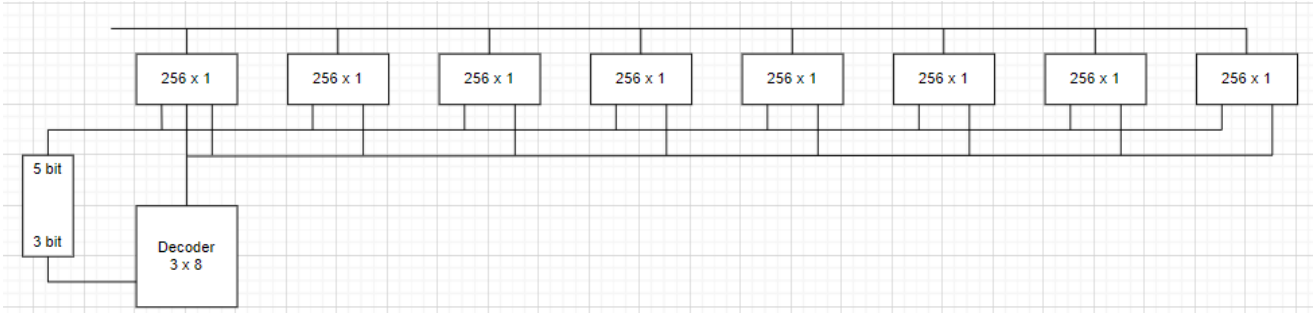
7.16 (Memories) Design:

a) 256K x 8 RAM using 265K x 1 RAM chips

Number of chips required: 256 x 8 / (256 x 1) = 8 chips 256 x 1

The requested size is: 256 x 8 = 2^8 x 8 → 8 bit address is necessary

Decoder size: To select 8 chips, 3 bits are needed (due to $2^3 = 8$) → Decoder 3 x 8

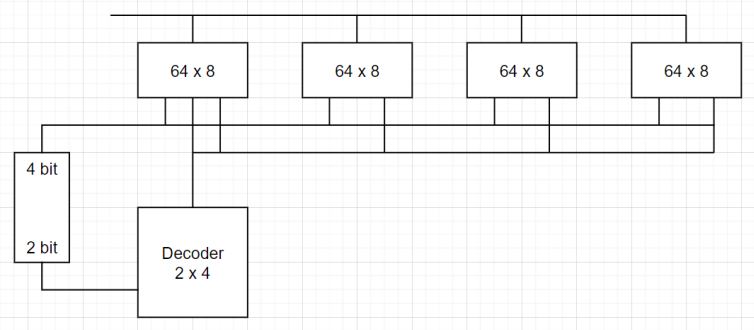


b) 64K x 32 RAM using 64K x 8 RAM chips

Number of chips required: 64 x 32 / (64 x 8) = 4 chip 64 x 8

The requested size is: 64 x 32 = 2^6 x 32 → 6 bit address is necessary

Decoder size: To select 4 chips, 2 bits are needed (due to $2^2 = 4$) → Decoder 2 x 4

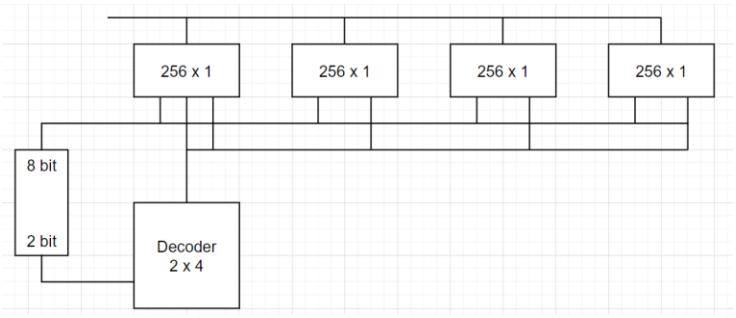


c) 1M x 1 RAM using 265K x 1 RAM chips

Number of chips required: 1M x 1 / 265k x 1 = 4 chip 256 x 1

The requested size is: 1M x 1 = 2^{10} →10 bit address is necessary

Decoder size: To select 4 chips, 2 bits are needed (due to $2^2 = 4$) → Decoder 2 x 4

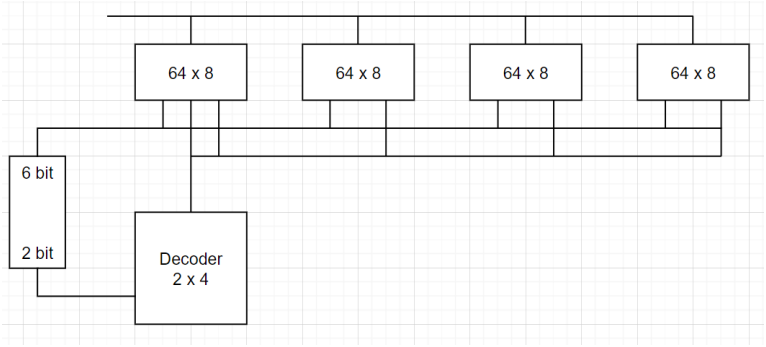


d) 265K x 8 RAM using 64K x 8 RAM chips

Number of chips required: 256 x 8 / (64 x 8) = 4 chip 64 x 8

The requested size is: 256 x 8 = 2^8 → 8 bit address is necessary

Decoder size: To select 4 chips, 2 bits are needed (due to $2^2 = 4$) → Decoder 2 x 4



7.17 (Stack) Design a push-down stack with 1K RAM that uses all (1024) words of the 1K RAM

Course code: CE118.P13
Lecturer: Ms. Ho Ngoc Diem

Name: Tran Ngoc Anh
Student code: 22520077