ECE 385

Fall 2020

Experiment #3

Lu Yicheng
Xiao Shuhong

1. Introduction

1.1. summary of high-level design

the circuit we build in lab 3 is an 8-bits logic processor consists of a register unit, a computation unit, a routing unit and a control unit.

the register unit is made up of two 4-bit shift registers used to hold the value.

the computation unit can realize 8 kinds of bitwise operation: AND, NAND, OR, NOR, XOR, XNOR, 1111 and 0000.

the routing unit choose where the result goes to, 4 kinds of mode can be chosen: remain unchanged, exchange A and B, keep A and result goes to B, keep B and result goes to A.

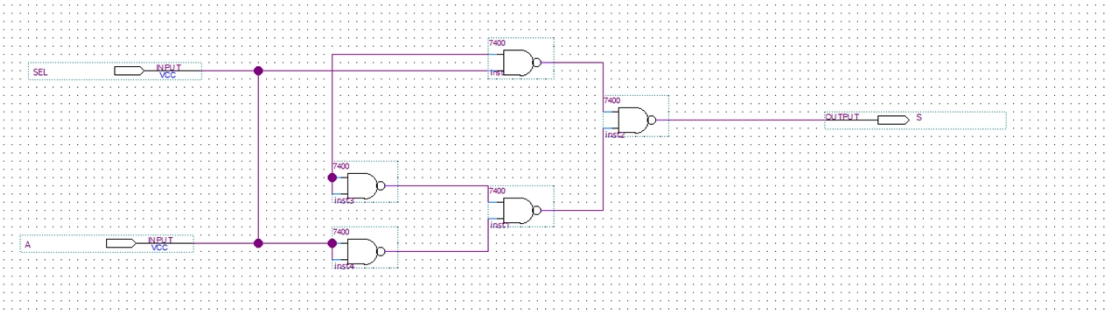
the control unit, based on a state machine, accept LoadA, LoadB, Excute and Clock as input, to decided whether any operation begin or keep unchanged.

1.2. answer to prelab question

A. we have input A and Sel, output S, define if Sel=1, S=A, otherwise, S=A'.

A	SEL	
0	0	1
0	1	0
1	0	0
1	1	1

S	SEL		
A		0	1
	0	1	0
	1	0	1



S=A'SEL'　+　ASEL

B. modular design can change the whole task into several smaller part, which is less complex with much simple function, it makes debugging much easier. also, it is very convenient if in later design the same function need to be realize in a different big task, we can reuse what we have built before to cut down development time.

2. Operation of the logic processor

2.1. switch sequence to load data

all our flip-flop change data at the rising edge. to load data into A, we set LoadA to 1, LoadB and Excute to 0, then the control unit(state machine) push mode choice 11 to our shift register to do parallel load, data from D0 to D3 then load into our A register; very similar, if we set LoadB to 1, LoadA and Excute to 0, mode choice 11, and data from D0 to D3 then load to B register.

2.2. switch sequence to initiate computation and routing operation

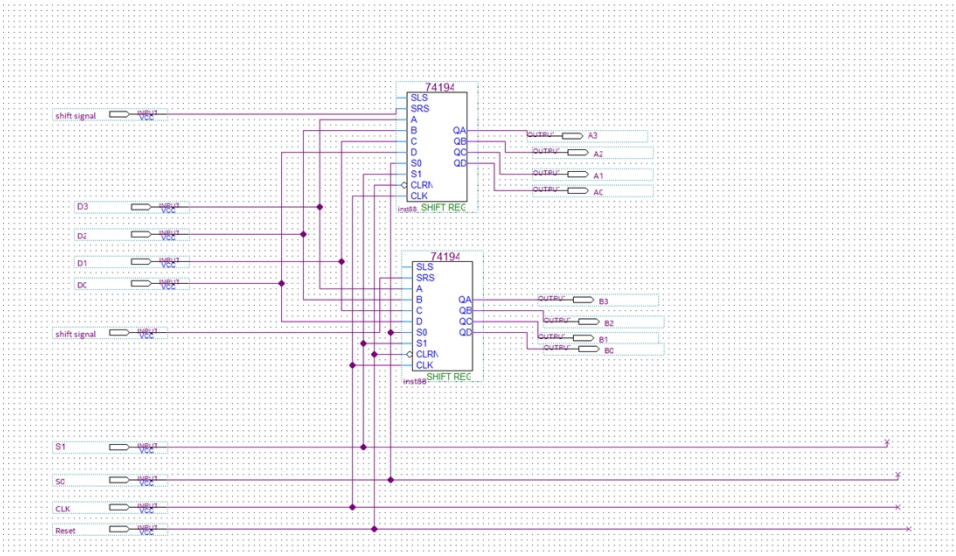
after load our value, to begin computation, we need to set Excute to 1, both LoadA and LoadB to 0, then, routing signal R1, R0 will decided which mode of routing to be used, our state machine will run four clock cycle to make sure just finish our four bit operation.

3. diagram description

3.1. written description

the circuit we build in lab 3 is an 8-bits logic processor consists of a register unit, a computation unit, a routing unit and a control unit.

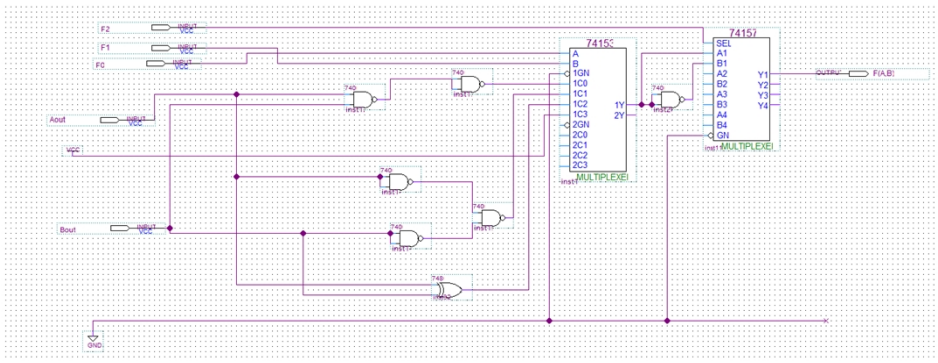
3.1.1. register unit



the register unit is made up of two 4-bit 74194 shift registers used to hold the

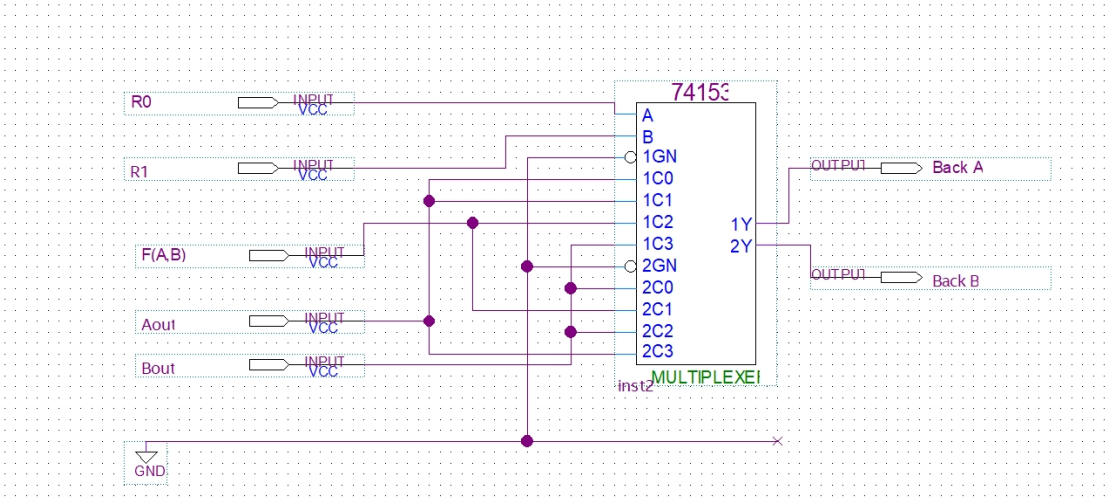
value. Input D3-D1 are used to load the value in when the load mode is chosen. S1 and S0 are mode choose signal to choose mode for 11 is parallel, 01 shifting and 00 hold. Two shifted signal are used for shifting, as data in register will only shift from left to right, we only connect SRS. Reset used to clear the mess and CLK used as our clock, all the change will only happen in the rising edges.

3.1.2. computation unit



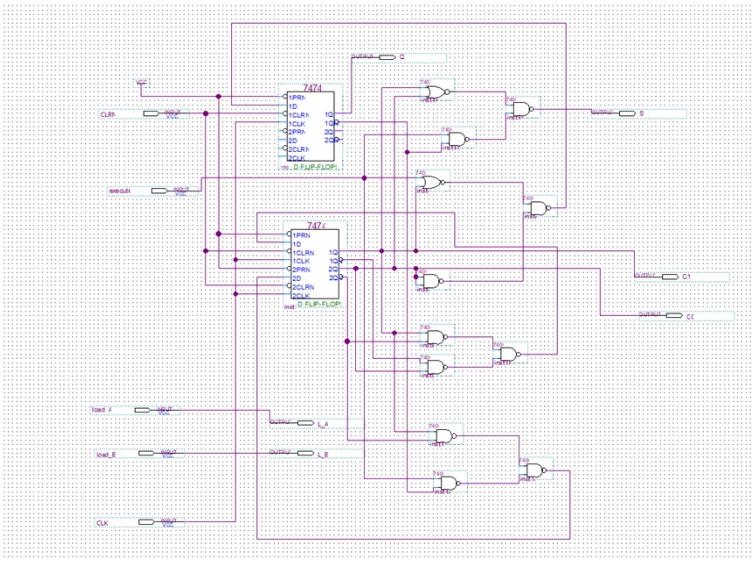
the computation unit can realize 8 kinds of bitwise operation: AND, NAND, OR, NOR, XOR, XNOR, 1111 and 0000. Input signal F2-F0 are used to choose the mod, input Aout and Bout shown in the figure are actually come from the least significance bit shift out from our register unit. The idea used here is first use a 4 to 1 mux to achieve four mode, and we find another 4 is just a negation of the achieved four, so another 2 to 1 mux are used here. Output here is the result from the chosen mode.

3.1.3. routing unit



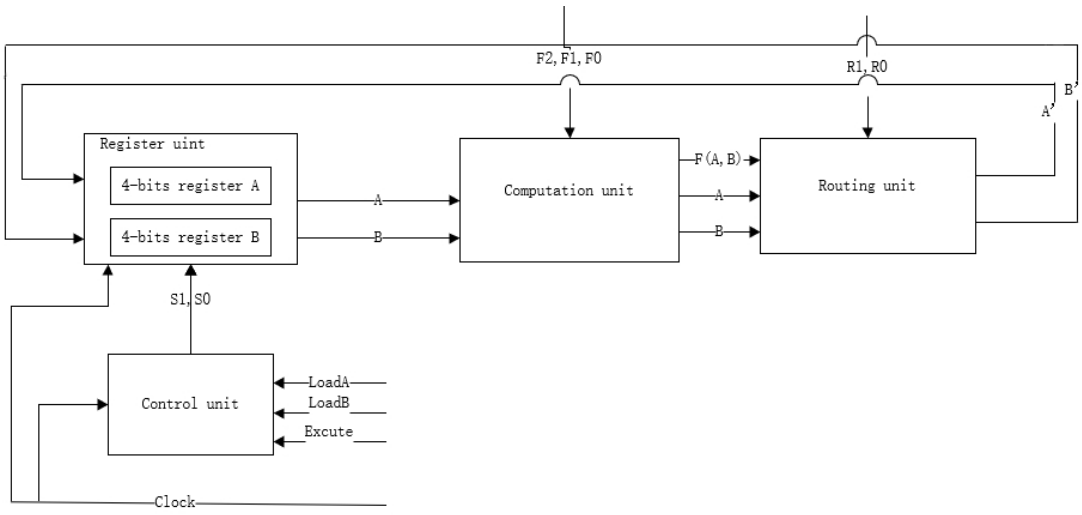
the routing unit choose where the result goes to, 4 kinds of mode can be chosen: remain unchanged, exchange A and B, keep A and result goes to B, keep B and result goes to A. Input signal R1 and R0 are used to select mode, F(A,B) from computation unit, A,B from register unit are used as input of this routing unit. Two 4 to 1 mux are used here to decide what to be store back to our register A and B.

3.1.4. control unit

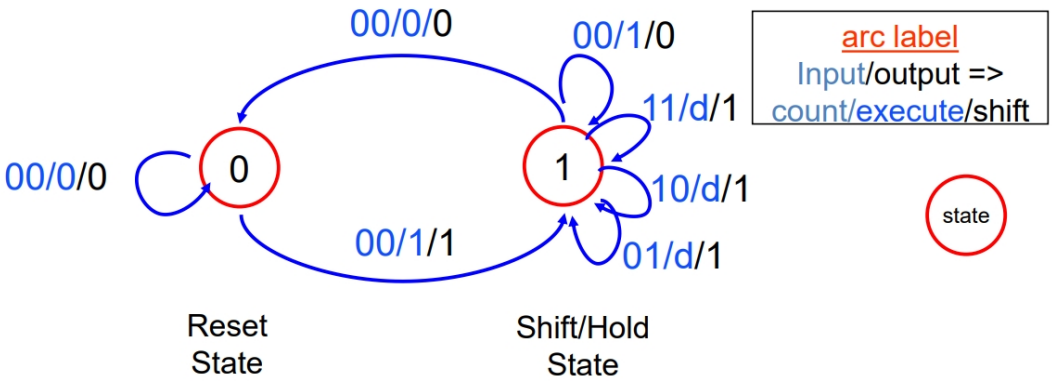


the control unit, based on a state machine, accept LoadA, LoadB, Excute and Clock as input, to decided whether any operation begin or keep unchanged. The control unit accept LoadA, LoadB and Excute as inputs, at any time at most one of the above three input should be set to 1, otherwise our control unit may not process correct output signal. one important thing for control unit is to achieve a mealy machine to achieve the need that for every execution, it should only run four loop to finish 4bit processing and then stop, so two 7474 D flip-flop chips (four filp flop total) are used here. The detailed state machine design will be shown in the state machine design part.

3.2. high-level diagram



3.3. state machine



3.3.1. Explicitly state

we use Mealy machine.

3.3.2. label each state

Give each state a meaningful name

The Reset sate and the Shift/hold state.

Specify the binary flip-flop values associated with each state.

The Reset sate: 0

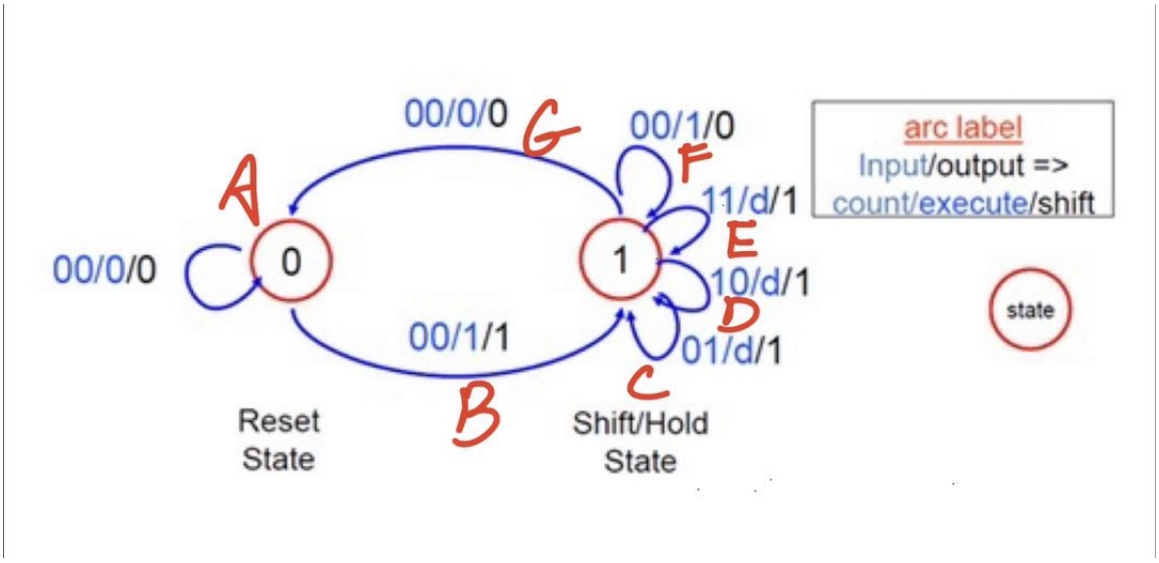
The Shift/hold state: 1

label the values of all meaningful outputs associated with each state.

EXCUTE	Q	C1C0	Input meaning	S	Q+	C1+C0+	Output meaning
0	0	00	Has no executing request	0	0	00	Stay in reset state and do nothing
1	0	00	Start execution. the first shift cycle.	1	1	01	Go to shift state to have the second shifting cycle.
d	1	01	the second shift cycle.	1	1	10	Stay in shift state, go to the third

							shift cycle.
d	1	10	the third shift cycle.	1	1	11	Stay in shift state, go to the fourth shift cycle.
d	1	11	the fourth shift cycle.	0	1	00	Finish executing
1	1	00	Finish executing, wait for executing signal to be 0	0	1	00	Finish executing, wait for executing signal to be 0
0	1	00	Finish executing, go to reset state	0	0	00	Return to reset state

3.3.2. label each state



EXCUTE	Q	C1C0	Input meaning	S	Q+	C1+C0+	Output meaning
0(A)	0	00	Has no executing	0	0	00	Stay in reset

			request				state and do nothing
1(B)	0	00	Start execution. the first shift cycle.	1	1	01	Go to shift state to have the second shifting cycle.
d(C)	1	01	the second shift cycle.	1	1	10	Stay in shift state, go to the third shift cycle.
d(D)	1	10	the third shift cycle.	1	1	11	Stay in shift state, go to the fourth shift cycle.
d(E)	1	11	the fourth shift cycle.	0	1	00	Finish executing
1(F)	1	00	Finish executing, wait for executing signal to be 0	0	1	00	Finish executing, wait for executing signal to be 0
0(G)	1	00	Finish executing, go to reset state	0	0	00	Return to reset state

4. Design steps taken and detailed circuit schematic diagram

4.1. procedure of the design step

4.1.1. Truth table and Kmap

Exec. Switch ('E')	Q	C1	C0	Reg. Shift ('S')	Q^+	$C1^+$	$C0^+$
0	0	0	0	0	0	0	0
0	0	0	1	d	d	d	D
0	0	1	0	d	d	d	D
0	0	1	1	d	d	d	D
0	1	0	0	0	0	0	0
0	1	0	1	1	1	1	0
0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	d	d	d	D
1	0	1	0	d	d	d	D
1	0	1	1	d	d	d	D
1	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	0

S	C1C0				
EQ		00	01	11	10
	00	0	d	d	d
	O1	0	1	1	1
	11	0	1	1	1
	10	1	d	d	d

S+=C1+C0+EQ'

Q	C1C0				
EQ		00	01	11	10
	00	0	d	d	d
	O1	0	1	1	1
	11	1	1	1	1
	10	1	d	d	d

Q+=C1+C0+E

C1	C1C0

EQ		00	01	11	10
	00	0	d	d	d
	O1	0	1	0	1
	11	0	1	0	1
	10	0	d	d	d

C1+=C1C0'+C1'C0

C0	C1C0				
EQ		00	01	11	10
	00	0	d	d	d
	O1	0	0	0	1
	11	0	0	0	1
	10	1	d	d	d

CO+=C1C0'+EQ'

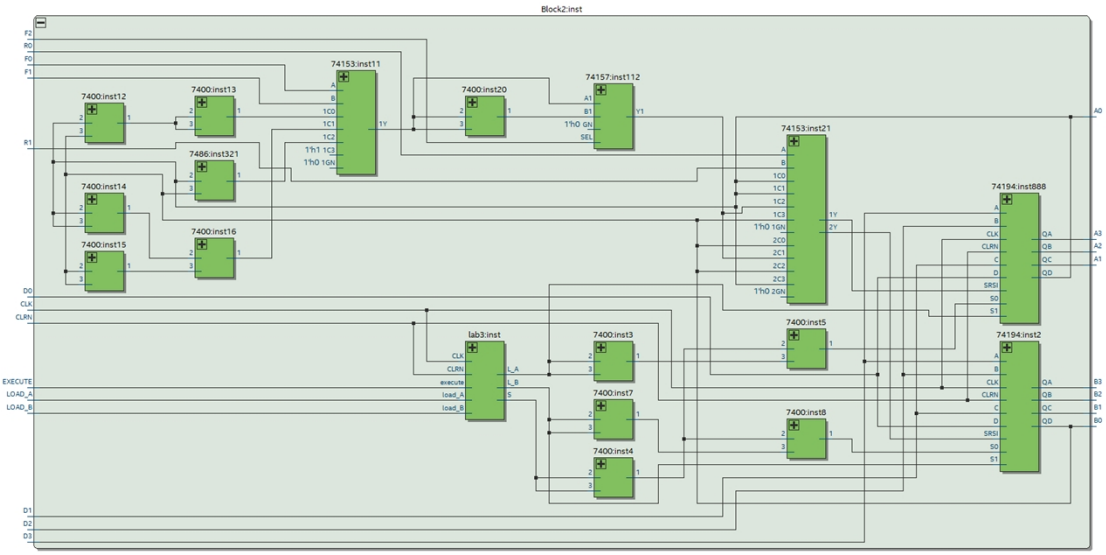
4.1.2. description of the design consideration taken

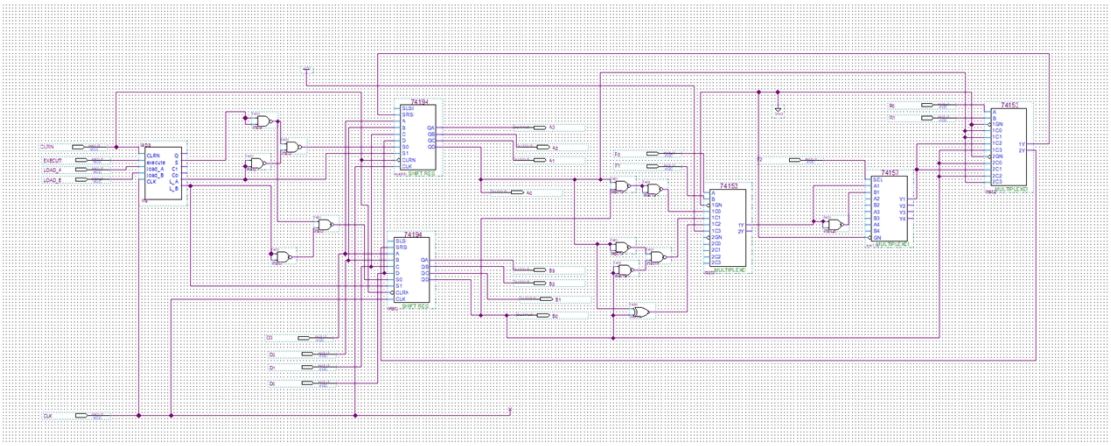
When designing the control unit, we have two options:

1, One is to use a two bits counter to represent number of shifting cycles. In this way, count signals are actually the input signals for the state machine. However, we find that we only have JK-negative trigger counter in the manual which means that C1C0 can only change at the falling edge while Q,S is actually at rising edge. To keep signals changes at the same time, we have to build a counter by ourselves(ie. Use the adder to build the counter, or just build the counter using gates and flip-flop). This cause the time and the circuit will be quite complex.

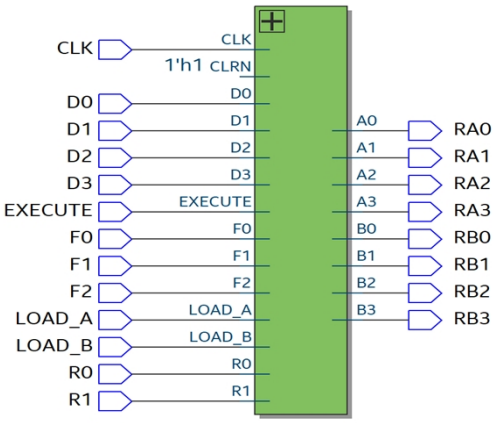
2, Another way is to relate C1+ and C0+ with current C1C0 and EQ. To be specific, we use two flip-flops to store C1C0 and the next value of C1C0 is depend on current C1C0 and EQ(C1+=C1C0'+C1'C0, CO+=C1C0'+EQ'). This combinational logic is obtained by drawing the truth table and kmap(list above).

4.2. Detailed Circuit Schematic

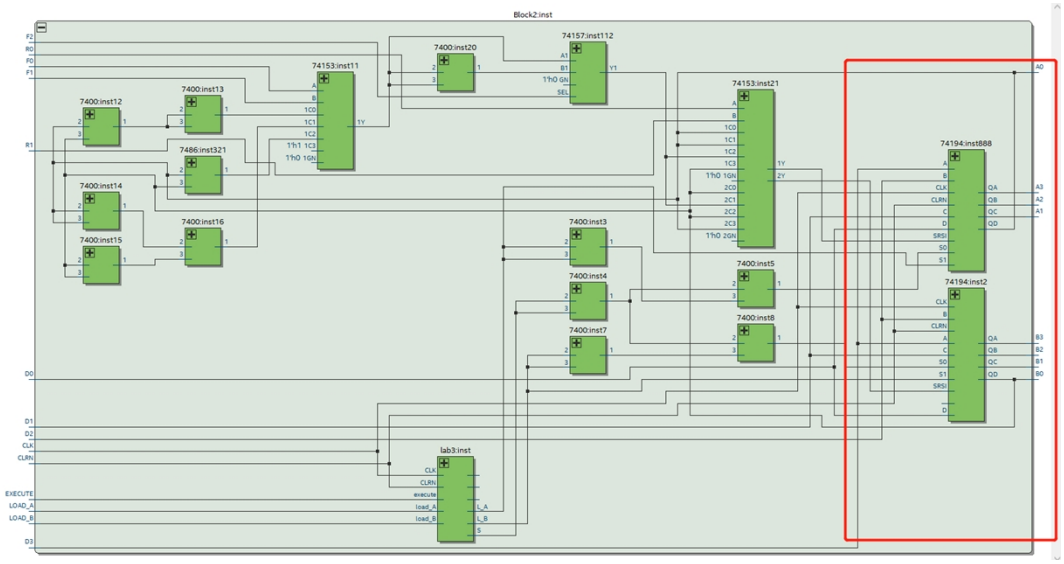


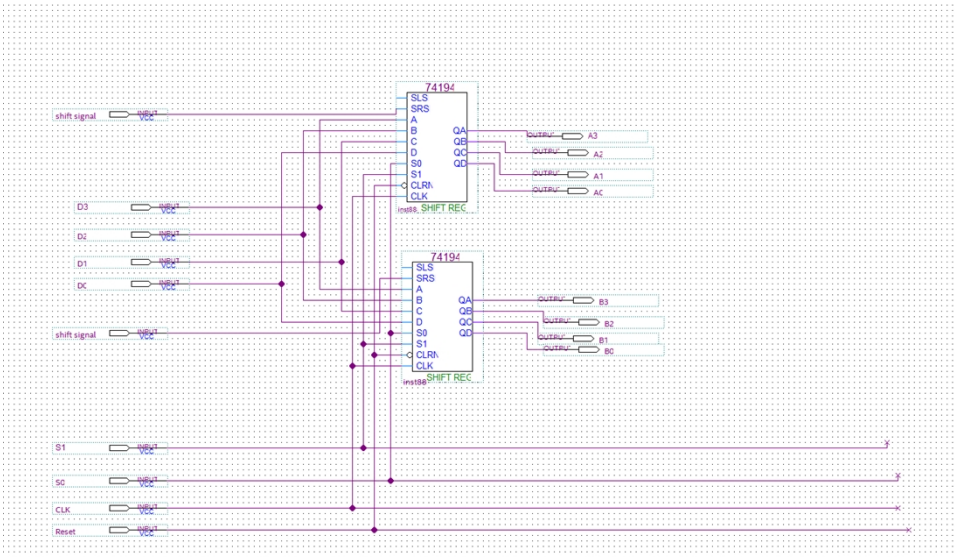


whole diagram

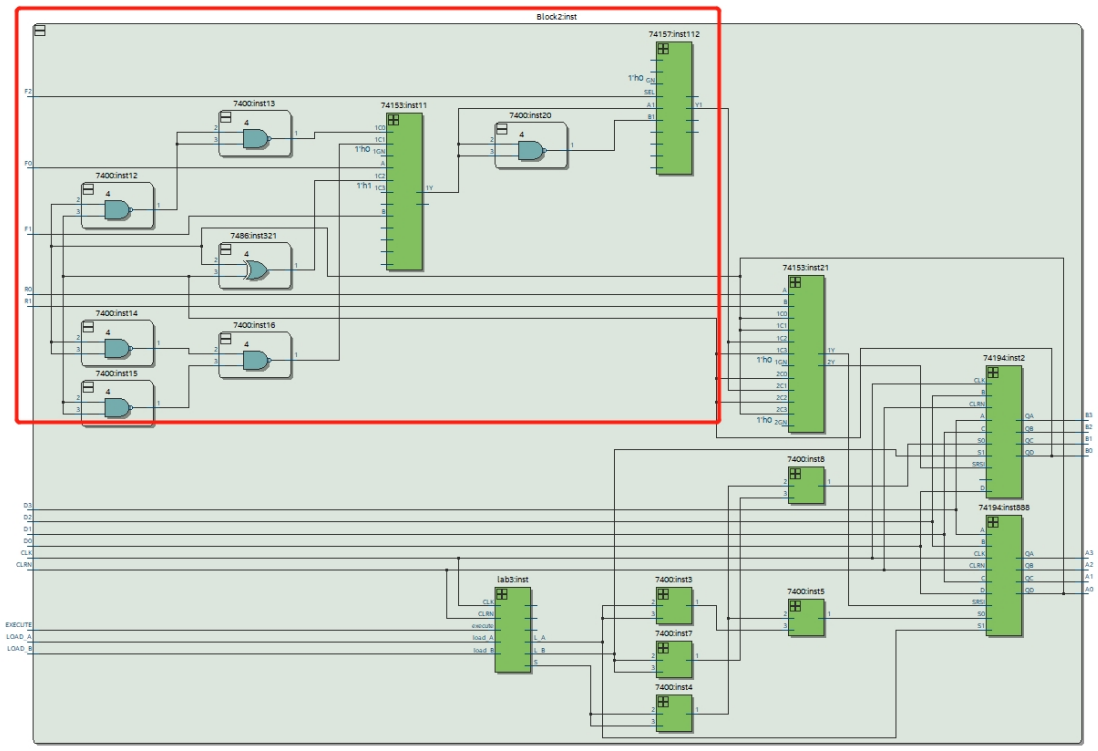


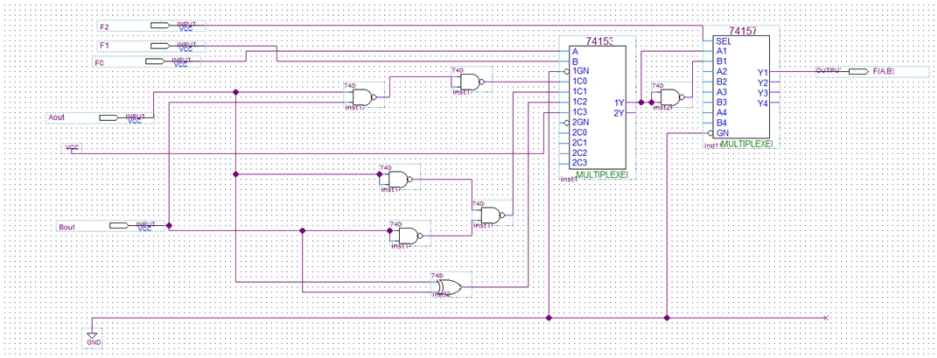
all input and output list above



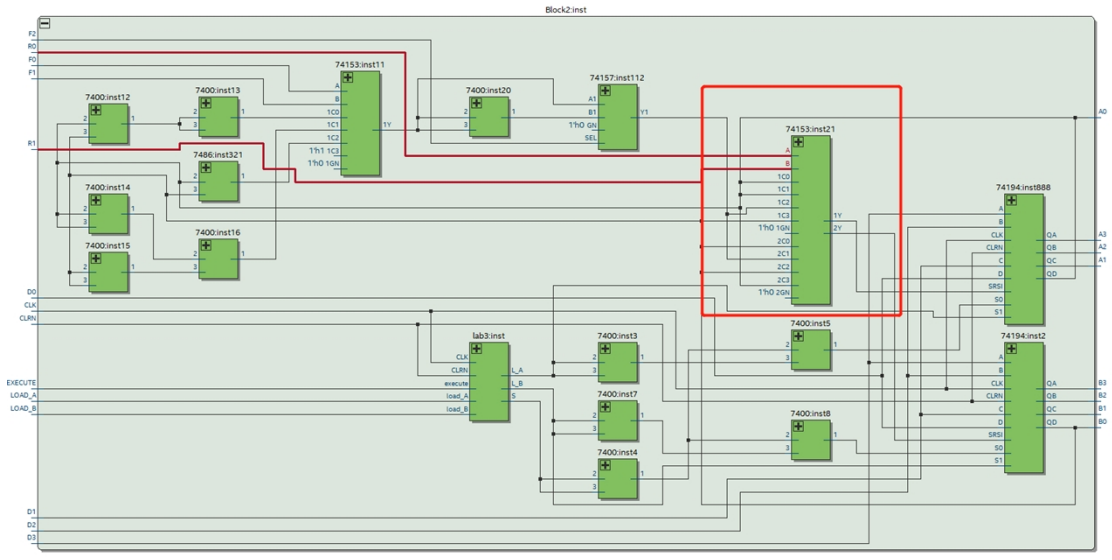


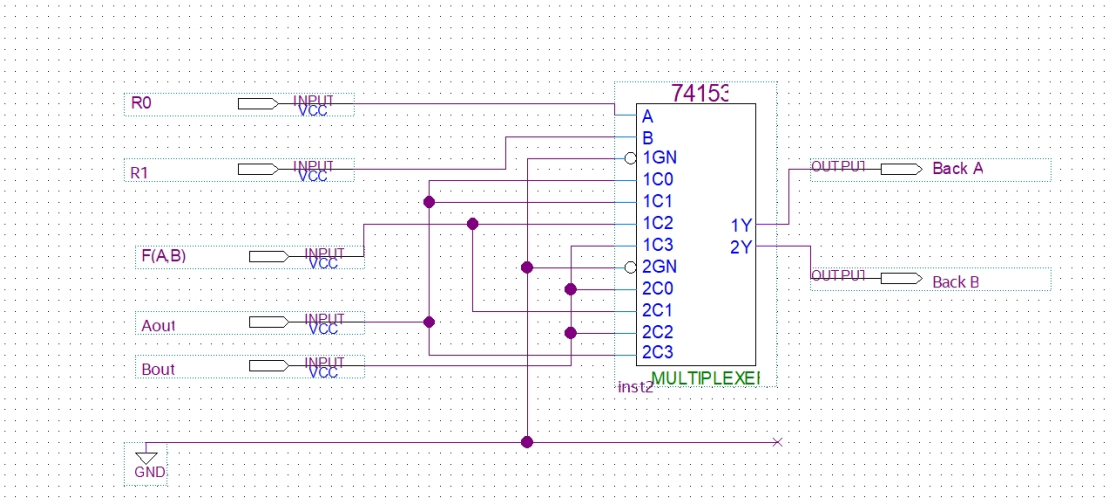
register unit part



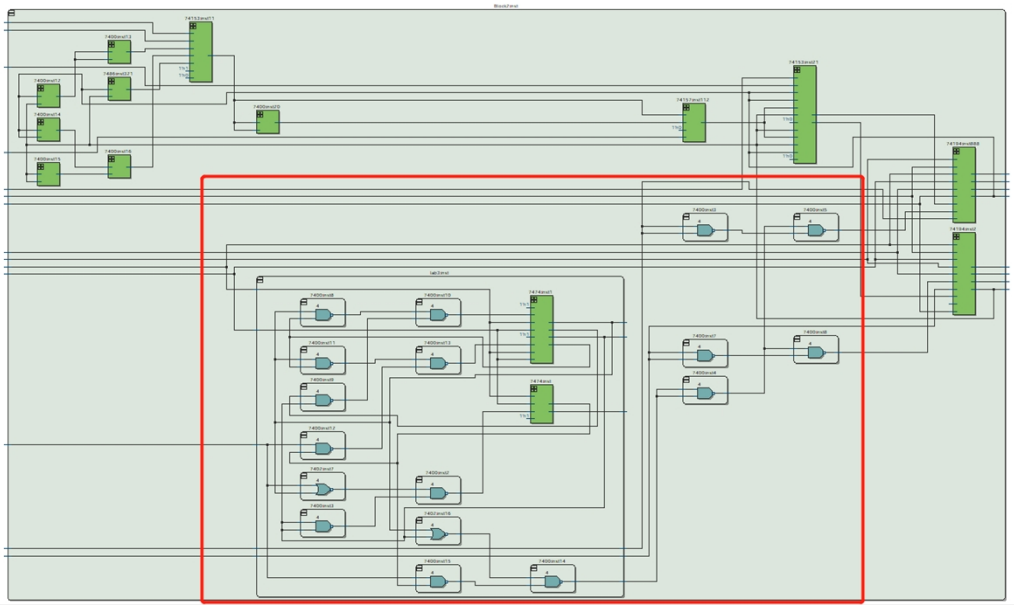


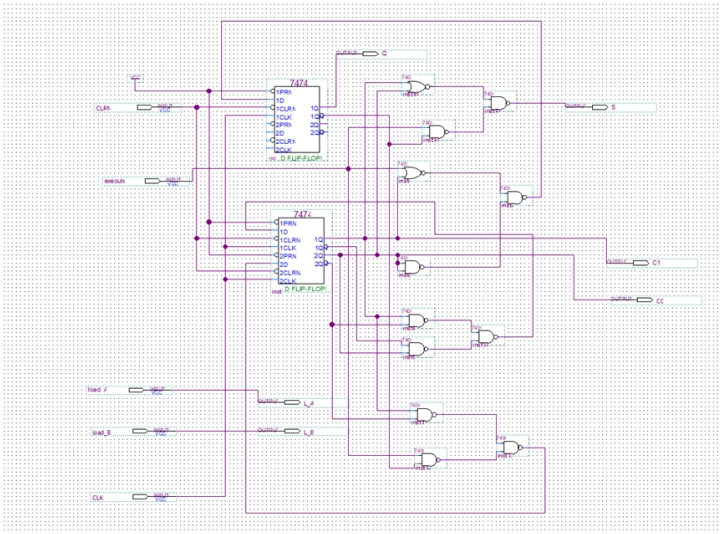
computation unit part





routing unit part





control unit part

5. all bugs encountered, and corrective measures taken The first bug happened because of the CLRN signal in 4-bit shift register (74194). We wrongly connect CLRN to gcc, which makes the register always keep 0000 in its data. As this signal will not appear in simulation, we take a long time to check where the trouble is and have to go through every block, and finally find this stupid work. and the way to fix it is just set the CLR as a input, so we can adjust it in our simulation, for example, set it high for the first several clock cycle to init register then keep it lower.

The second bug happens when connect 4 to 1 mux, as this mux have two bit select signal, in the beginning we wrongly swop the connection of selection signal A and B, so when doing simulation, we find what we finally get through routing unit are different from what describe in the table. So, we just swop them to correct.

6. Conclusion

6.1. Summarize

In this lab we design a 2x4 bit processor, the idea of modular help us to divide the whole project into four independent part: register unit, computation unit, routing unit and control unit to decrease the complexity and help debug. A mealy state machine are used in this design and help us be more familiar with both Moore and Mealy machine. Also, we gained some idea about how cpu works in real word.

6.2. post-lab question

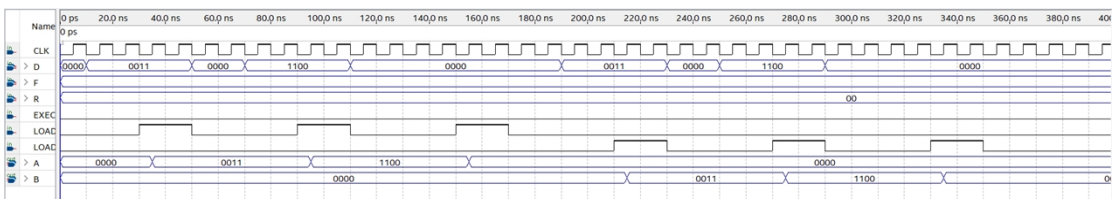
a. the idea of modular help us to debug more simply, as for every small part, we ca debug them independent and this is much easier than enhance the whole large design in one time.

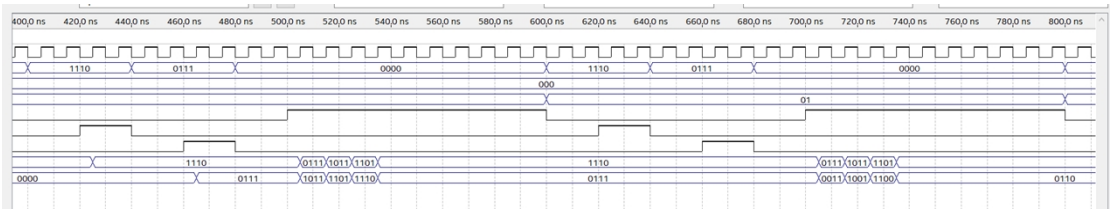
b. the different between Moore machine and Mealy machine is that the result of Moore machine only depends on the current state while for Mealy machine, both current state and input will affect result.

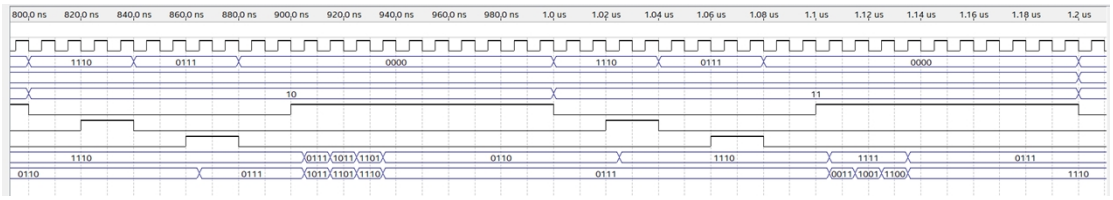
So for lab3 design, we finally choose Mealy machine to design our control logic, the reason is using Mealy machine, we can reduce the number of state, as Shift and Hold state can be combined so for two state we can only use 1 bit to represent it. while for state machine we need at least 2 bits to represent reset, shift and halt.

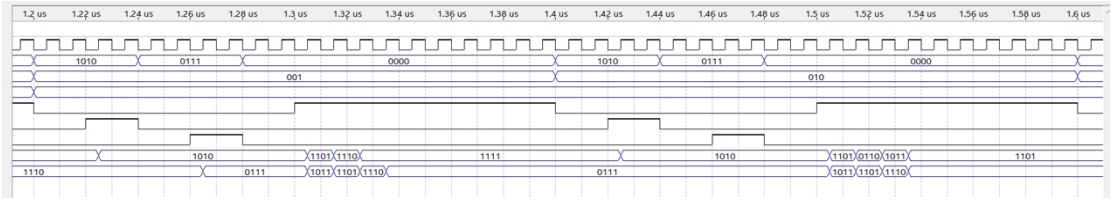
However, another problem for Mealy machine is that it is not so straightforward as Moore machine, sometimes we can hardly design using Mealy.

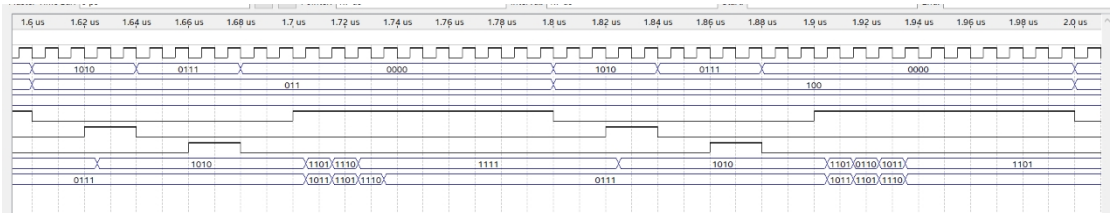
7. Simulation

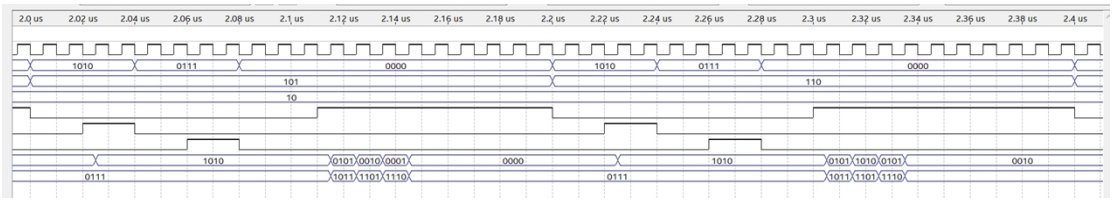


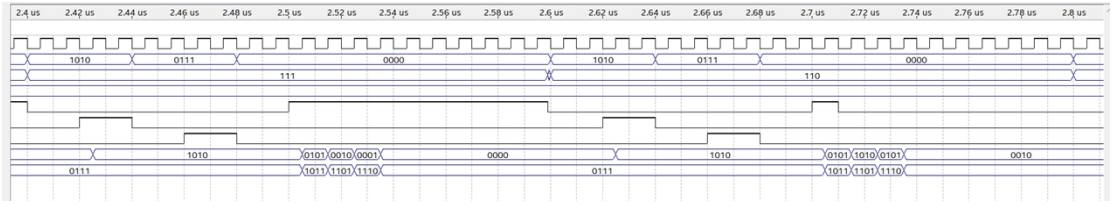

400,0 ns 420,0 ns 440,0 ns 460,0 ns 480,0 ns 500,0 ns 520,0 ns 540,0 ns 560,0 ns 580,0 ns 600,0 ns 620,0 ns 640,0 ns 660,0 ns 680,0 ns 700,0 ns 720,0 ns 740,0 ns 760,0 ns 780,0 ns 800,0 ns
1110 0111 0000 1110 0111 0000
000
01
1110 0111 1011 1101 1110 0111 1011 1101
0000 0111 1011 1101 1110 0111 0011 1001 1100 0110

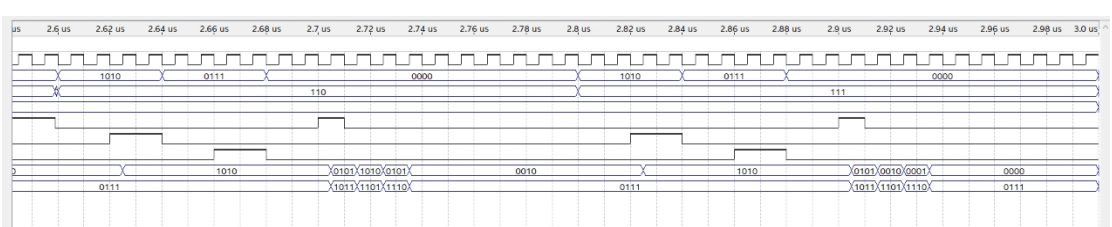

800,0 ns 820,0 ns 840,0 ns 860,0 ns 880,0 ns 900,0 ns 920,0 ns 940,0 ns 960,0 ns 980,0 ns 1,0 us 1,02 us 1,04 us 1,06 us 1,08 us 1,1 us 1,12 us 1,14 us 1,16 us 1,18 us 1,2 us
1110 0111 0000 1110 0111 0000
10
11
1110 0111 1011 1101 0110 1110 1111 0111
0110 0111 1011 1101 1110 0111 0011 1001 1100 1110


1,2 us 1,22 us 1,24 us 1,26 us 1,28 us 1,3 us 1,32 us 1,34 us 1,36 us 1,38 us 1,4 us 1,42 us 1,44 us 1,46 us 1,48 us 1,5 us 1,52 us 1,54 us 1,56 us 1,58 us 1,6 us
1010 0111 0000 1010 0111 0000
001
010
1010 1101 1110 1111 1010 1101 0110 1011 1101
1110 0111 1011 1101 1110 0111 1011 1101 1110


1,6 us 1,62 us 1,64 us 1,66 us 1,68 us 1,7 us 1,72 us 1,74 us 1,76 us 1,78 us 1,8 us 1,82 us 1,84 us 1,86 us 1,88 us 1,9 us 1,92 us 1,94 us 1,96 us 1,98 us 2,0 us
1010 0111 0000 1010 0111 0000
011
100
1010 1101 1110 1111 1010 1101 0110 1011 1101
0111 1011 1101 1110 0111 1011 1101 1110


2,0 us 2,02 us 2,04 us 2,06 us 2,08 us 2,1 us 2,12 us 2,14 us 2,16 us 2,18 us 2,2 us 2,22 us 2,24 us 2,26 us 2,28 us 2,3 us 2,32 us 2,34 us 2,36 us 2,38 us 2,4 us
1010 0111 0000 1010 0111 0000
101
110
10
1010 0101 0010 0001 0000 1010 0101 1010 0101 0010
0111 1011 1101 1110 0111 1011 1101 1110


2,4 us 2,42 us 2,44 us 2,46 us 2,48 us 2,5 us 2,52 us 2,54 us 2,56 us 2,58 us 2,6 us 2,62 us 2,64 us 2,66 us 2,68 us 2,7 us 2,72 us 2,74 us 2,76 us 2,78 us 2,8 us
1010 0111 0000 1010 0111 0000
111
110
1010 0101 0010 0001 0000 1010 0101 1010 0101 0010
0111 1011 1101 1110 0111 1011 1101 1110


2,6 us 2,62 us 2,64 us 2,66 us 2,68 us 2,7 us 2,72 us 2,74 us 2,76 us 2,78 us 2,8 us 2,82 us 2,84 us 2,86 us 2,88 us 2,9 us 2,92 us 2,94 us 2,96 us 2,98 us 3,0 us
1010 0111 0000 1010 0111 0000
110
111
1010 0101 1010 0101 0010 1010 0101 0010 0001 0000
0111 1011 1101 1110 0111 1011 1101 1110 0111