LAB REPORT

Report One - Expandable ALU and shifter register

YinHang Kwok 800831787
Voltaire Medina 800861766

ITCS - 3181/L FALL 2016

October 6, 2016

Introduction & Purpose

The Arithmetic Logic Unit (ALU) is a digital circuit used to process arithmetic and logic operations. ALU is an important part of control unit inside the CPU. ALU performs basic arithmetic and logic operation where all information in a computer is stored in the form of binary number (0,1). Also, ALU gets input and output from processor controller, main memory and I/O devices.

After user input some information into computer, these information and instructions travel among electronic path call bus. These information and instructions have opcode and operands. The opcode will tell what function ALU have to do. The operands are some information, such as address of data in memory and data. Transistors are contained in the ALU to switch and manipulate binary numbers.

This lab made us use our previously made components from labs in the past (shifter, adder-subtracter, etc.) and newly made components (4-Pass A, 4-NOT,4-AND gate), and knowledge about the ALU and its components learned from the lecture portion of this class to piece together a functional 4-bit-expandable ALU. Then, after building the ALU component, we would add a shifter register functionality to create a 4-bit-ALU-shifter register.

We both were involved with coming up with the design of the ALU and refining it so that the output of all components were logical, and matched our expected truth tables. Our designs for the ALU on paper were very similar and had all the components needed as well as the outputs and the inputs.

Piecing together the 4-bit-ALU-shifter was quite easy as it was just a small additional component added to the schematic. The whole lab was just a matter of connecting the right inputs with the right outputs, given the fact that our previously made components were functional and correctly structured. However, in order to do that we needed to know how the ALU works and understand the flow of data from component to component.

We had some technical issues with matching the correct inputs of certain components due to the complexity of the number of components, for example were running out of space when connecting nets because the design had so many connections, and it made our design look a bit chaotic. We did clean our schematic to make it more presentable and to make debugging much easier as the nets were easy to follow.

We also came across some logical problems with the 3 inputs needed to determine the functionality of the ALU because at the time we did not realize that the F0 input was the same as CIN. Another problem we encountered is the mismatching of some input and output values of the components, that some of our inputs/outputs of some of our components were not consistent. For example, for one component out outputs were S4, S3, S2, S1 and the inputs for another component that used the outputs of the component mentioned before was Q1, Q2, Q3, Q4 (The output/input names are reversed and named differently). This made it quite confusing for us to match our nets in our schematics.

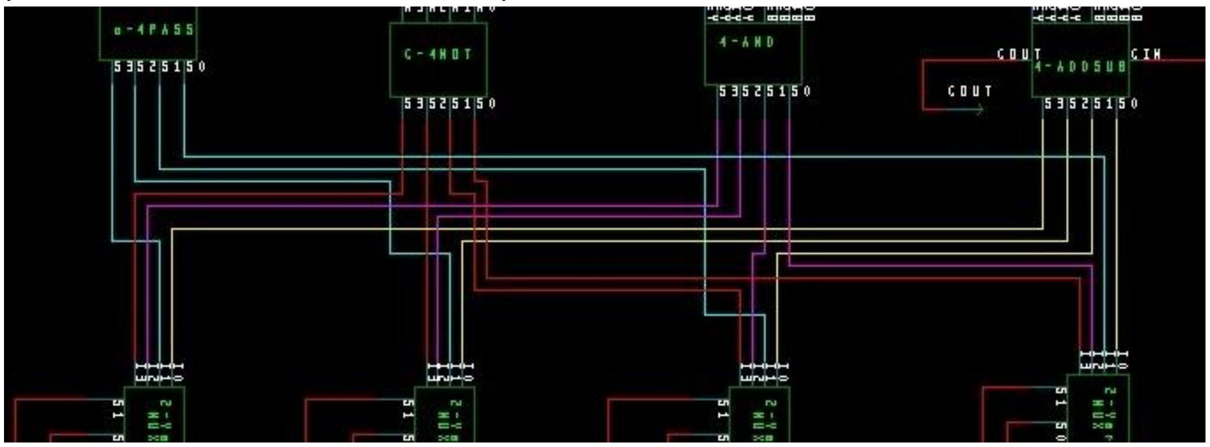
The design of ALU / Shift- register

This lab's objective is to design a 4-bit expandable ALU and combine it with shift register to make a functional 4-bit ALU shifter.

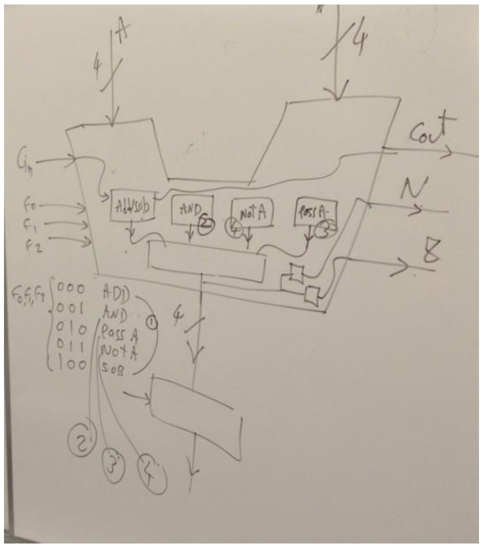
First, our 4-bit expandable ALU design contains 11 inputs. They are A(A0,A1,A2,A3), B(B0,B1,B2,B3), F(F1,F2) and CIN. F is selector input to control the operation of ALU. Since F0 are same as CIN, we use CIN to replace F0.

Second, our ALU contain four main functions, such as 4-bit adder and subtracter, 4-AND, 4-NotA and 4-PassA. The Adder-subtracter computes the value of (+/-) from two 2-bit binary numbers (input A_n and B_n where n is 0,1,2,3). The 4-AND gate passes the value of 4 AND gates with 4 inputs of A and B. The 4-NOTA passes the inverted 4 values of A. The 4-PassA is designed to let each A input passes through 2 not gates in order to keep to value unchanged. Also, there are 4 2-var-MUX to select the output form specific function. The selection process bases on the F input (select lines). In addition, there is a 4-NotOR gate to compute output Z.

There is main design idea in the connection between 4 operation functions (add/sub, 4notA,4AND and 4PassA) and 4 2-Var-MUXs.



This design pattern is basing on the table of operation control. For example operation is AND when F values are 0, 0 ,0 and operation is PassA when F values are 0,1,0. Base on the required table from the lecture, the outputs from the add/sub should connect to each I0 in each 2 Var-MUX. The outputs of 4PassA should connect to each I1 in each 2 Var-MUX. The outputs of 4AND should connect to each I2 in each 2 Var-MUX. The outputs of 4NOTA should connect to each I3 in each 2 Var-MUX.



The design requirement of ALU. It showed the 000 for AND, 001 for AND, 010 for PassA, 011 for NOTA.

When it comes to the output of ALU, there are 4 types of output, such as Z, N,COUT and Q(Q0,Q1,Q2,Q3). Z and N are used to control the micro seq. logic component. The COUT is the output that it carry out the output of the adder-subtracter. The Q outputs are the 4-bit output of the 4-bit ALU.

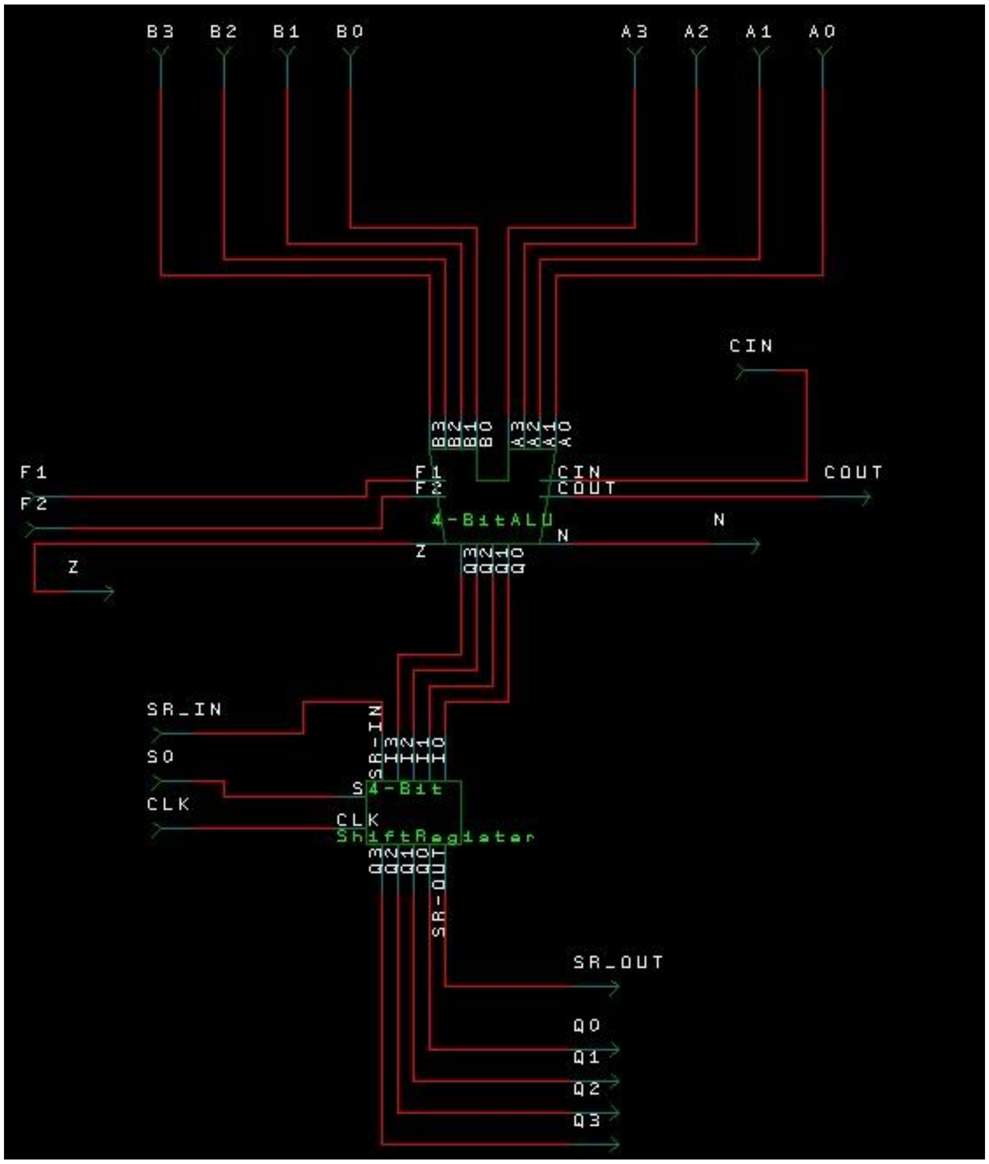
Our schematic uses inputs of A and B as inputs of each of the 4 components used in the ALU. Pass A and 4-not-A will use only the values of A, and 4AND and the adder-subtractor will use both the values of A and B. Those values will then be sent to the four 2 variable multiplexers (one of the 4 outputs of the 4 components will an input of each of the 4 multiplexers) of whom's output will be determined from the input values from the 3 inputs of F (CIN, F1, F2). The values of Q is then determined as the output of the selected component of one of the 4 logic/arithmetic components. N is computed from the Q3 output using a double not gate. And Z is computed using the 4-not-AND gate.

When it comes to the ALU shift register, it is the combined with ALU and shift register. There are 3 inputs for shift register, such as SR_IN, CLK and S0. SR_IN is controlling the latch function for the shifter. The CLK generates the clock signal to control the flip-flops process in latch. The S0 is the select line to control the latch function.

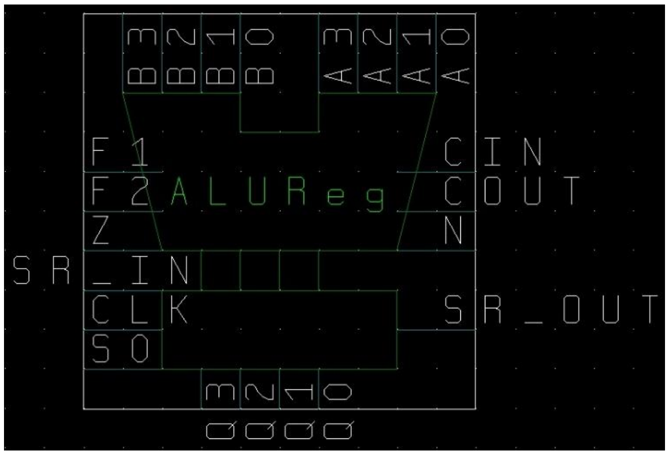
This schematic follows the same flow as the 4-bit ALU but with the added shifter component. The 4-bit Q output of the ALU will be used as the inputs of the shifter where the shift is determined from the S, and SR_IN. The CLK is then used for the shifter's memory aspect. The SR_OUT value and the Q outputs of the shift-ALU will then be used late for the register component.

<u>Symbol and Schematic design of ALU and Shift-register</u>

Schematic of ALU/Register



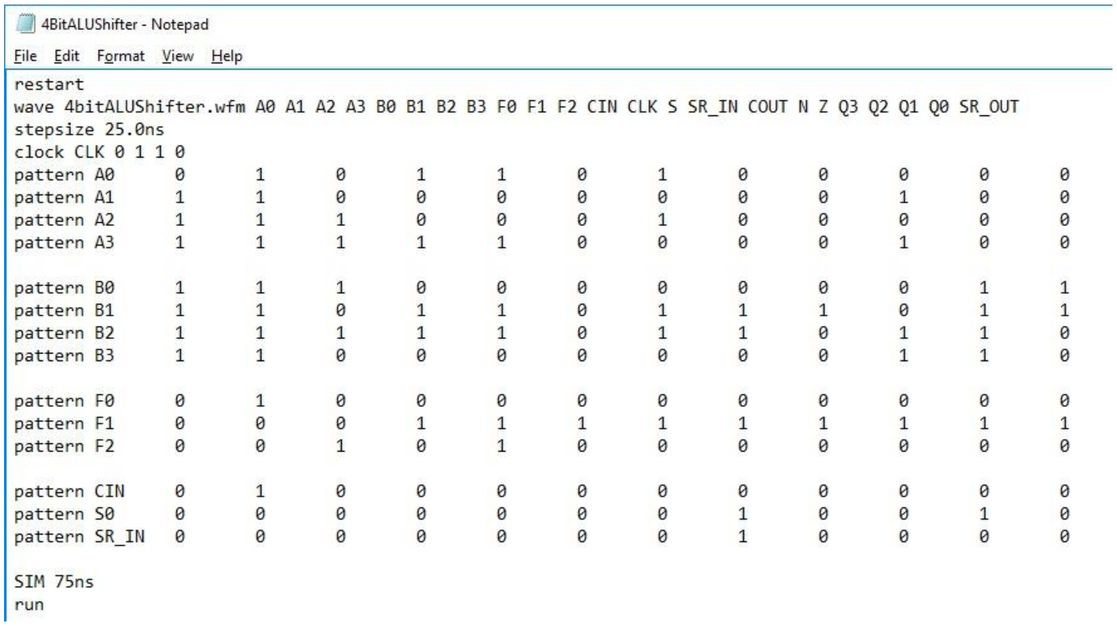
Symbol



expected results table

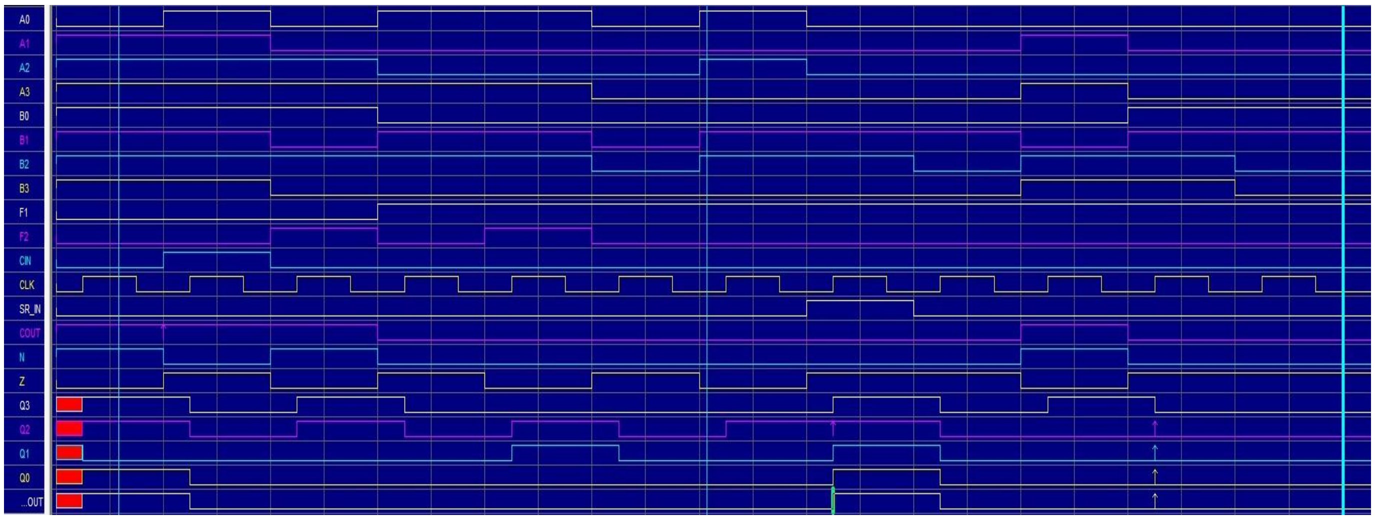
A0	A1	A2	A3	B0	B1	B2	B3	F0	CIN	SR_IN	COUT	N	Z	Q3	Q2	Q1	Q0	SR)OUT
0	1	1	1	1	1	1	1	0	0	0	1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	0	1	1	0	1	1	0	0	0
1	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0

command file



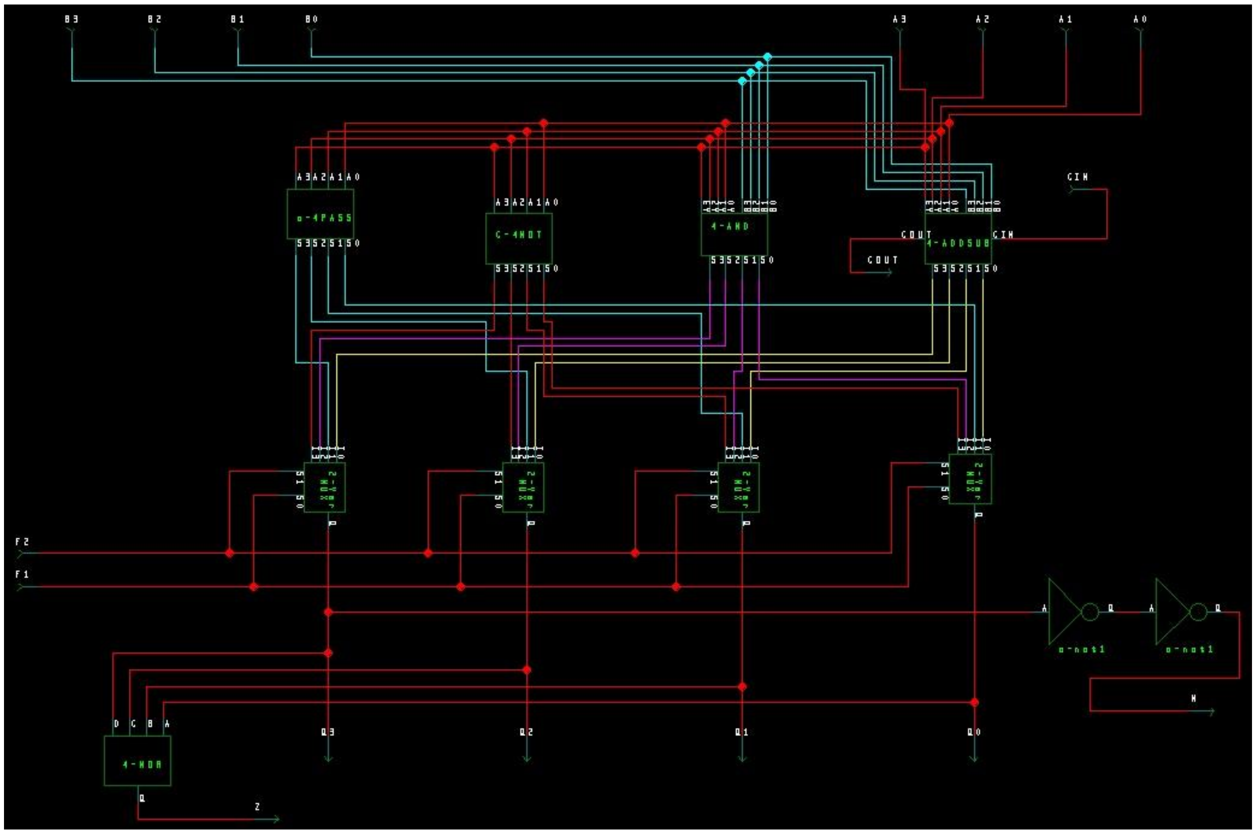
```
restart
wave 4bitALUShifter.wfm A0 A1 A2 A3 B0 B1 B2 B3 F0 F1 F2 CIN CLK S SR_IN COUT N Z Q3 Q2 Q1 Q0 SR_OUT
stepsize 25.0ns
clock CLK 0 1 1 0
pattern A0      0       1       0       1       1       0       1       0       0       0       0       0
pattern A1      1       1       0       0       0       0       0       0       0       1       0       0
pattern A2      1       1       1       0       0       0       1       0       0       0       0       0
pattern A3      1       1       1       1       1       0       0       0       0       1       0       0

pattern B0      1       1       1       0       0       0       0       0       0       0       1       1
pattern B1      1       1       0       1       1       0       1       1       1       0       1       1
pattern B2      1       1       1       1       1       0       1       1       0       1       1       0
pattern B3      1       1       0       0       0       0       0       0       0       1       1       0

pattern F0      0       1       0       0       0       0       0       0       0       0       0       0
pattern F1      0       0       0       1       1       1       1       1       1       1       1       1
pattern F2      0       0       1       0       1       0       0       0       0       0       0       0

pattern CIN     0       1       0       0       0       0       0       0       0       0       0       0
pattern S0      0       0       0       0       0       0       0       1       0       0       1       0
pattern SR_IN   0       0       0       0       0       0       0       1       0       0       0       0

SIM 75ns
run
```

simulation result

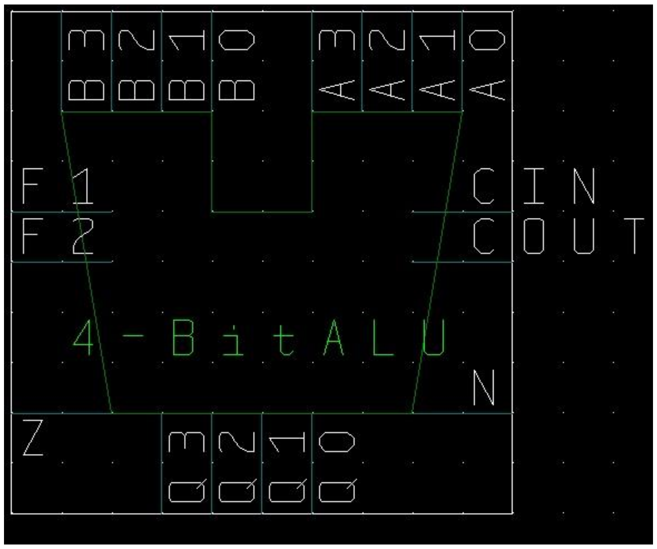


Symbol and Schematic design of ALU

Schematic



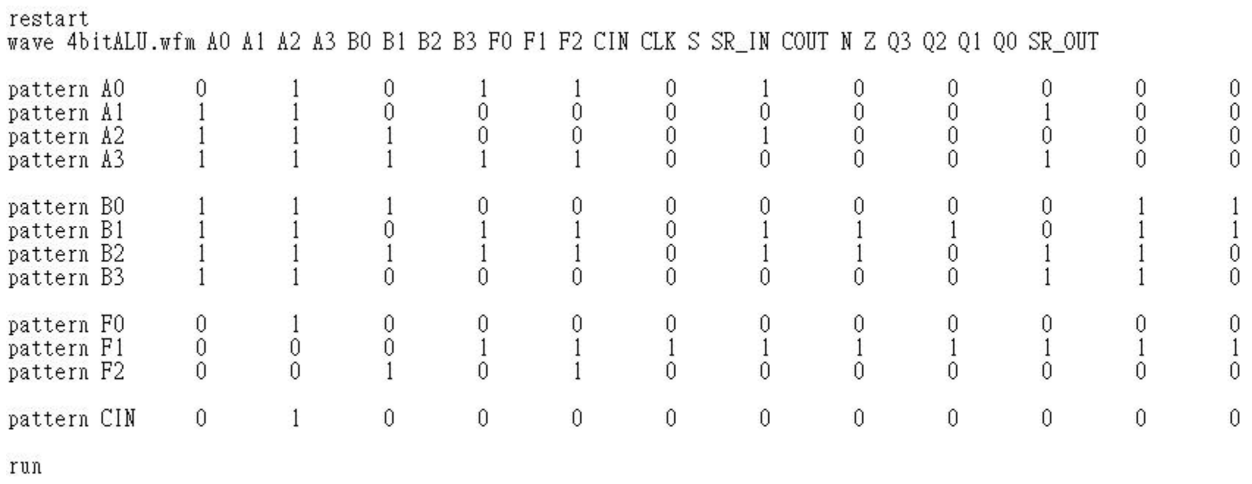
Symbol



expected results table

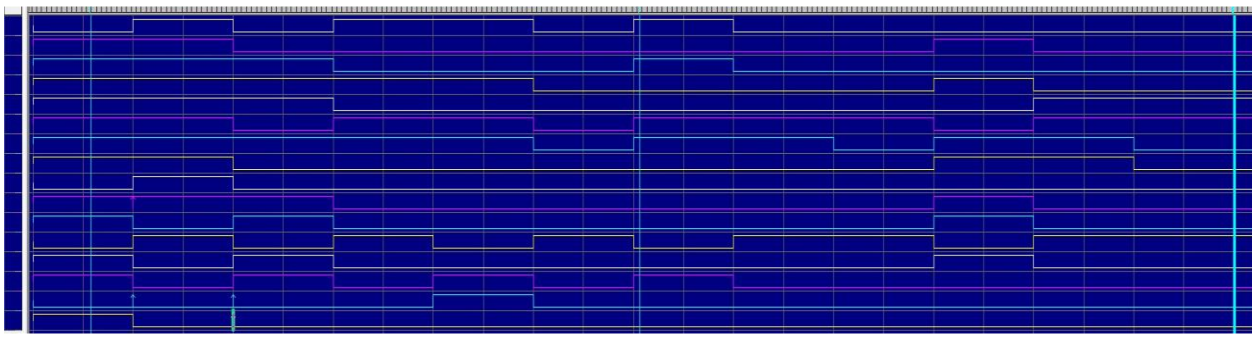
A0	A1	A2	A3	B0	B1	B2	B3	F0	CIN	N	Z	Q3	Q2	Q1	Q0
0	1	1	1	1	1	1	1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
0	0	1	1	1	0	1	0	0	0	1	0	1	1	0	0
1	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	1	0	0	1	0	1	0	0	0
0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0

command file



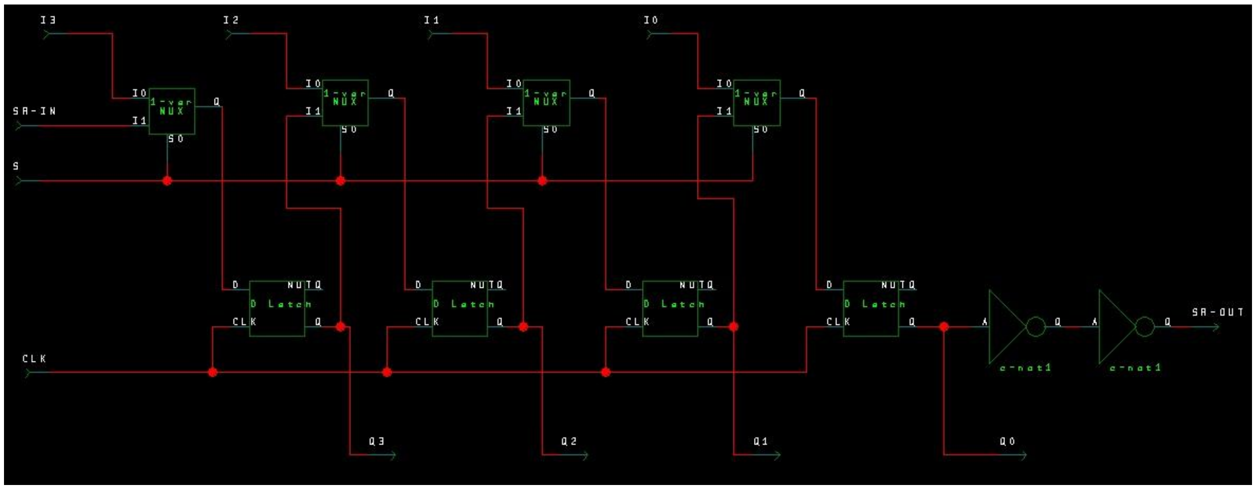
```
restart
wave 4bitALU.wfm A0 A1 A2 A3 B0 B1 B2 B3 F0 F1 F2 CIN CLK S SR_IN COUT N Z Q3 Q2 Q1 Q0 SR_OUT

pattern A0    0   1   0   1   1   0   1   0   0   0   0   0
pattern A1    1   1   0   0   0   0   0   0   0   1   0   0
pattern A2    1   1   1   0   0   0   1   0   0   0   0   0
pattern A3    1   1   1   1   1   0   0   0   0   1   0   0

pattern B0    1   1   1   0   0   0   0   0   0   0   1   1
pattern B1    1   1   0   1   1   0   1   1   1   0   1   1
pattern B2    1   1   1   1   1   0   1   1   0   1   1   0
pattern B3    1   1   0   0   0   0   0   0   0   1   1   0

pattern F0    0   1   0   0   0   0   0   0   0   0   0   0
pattern F1    0   0   0   1   1   1   1   1   1   1   1   1
pattern F2    0   0   1   0   1   0   0   0   0   0   0   0

pattern CIN   0   1   0   0   0   0   0   0   0   0   0   0

run
```

simulation result

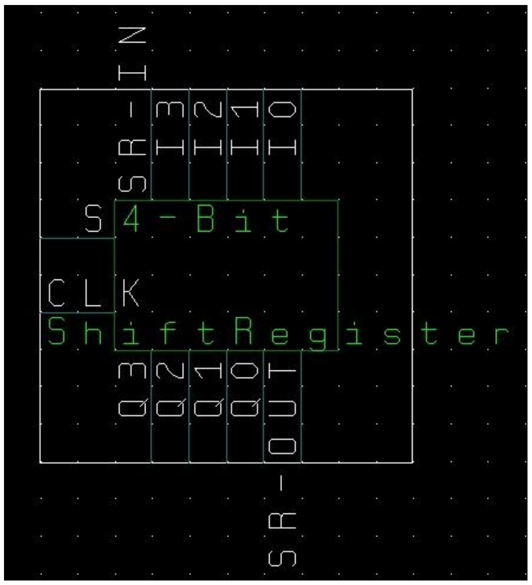


Symbol and Schematic design of shift-register

Schematic



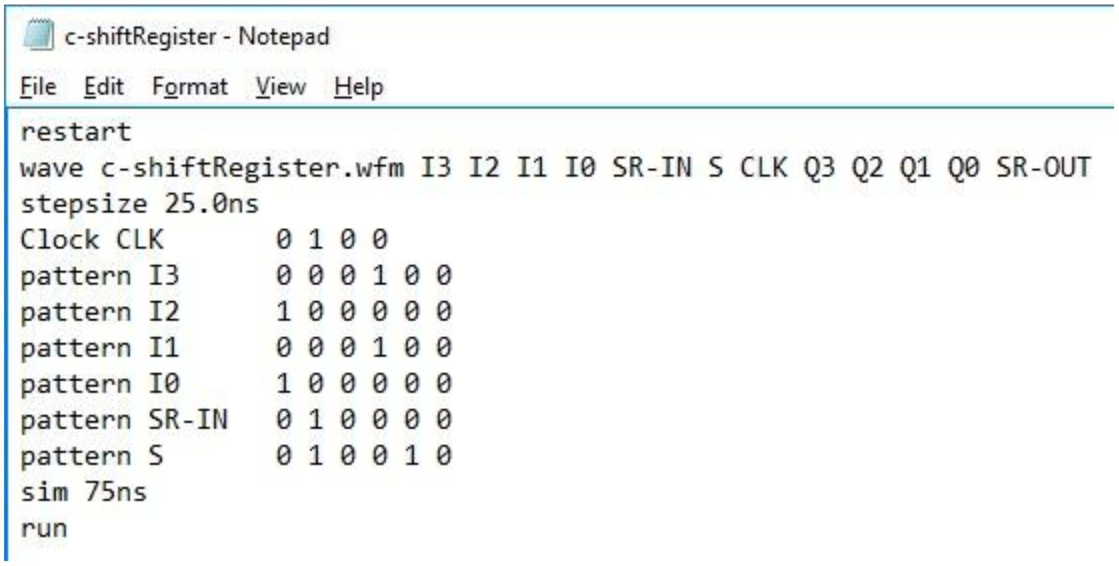
Symbol



expected results table

I0	I1	I2	I3	S	SR_IN	S3	S2	S1	S0	SR_OUT
1	0	1	0	0	0	0	1	0	1	1
0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

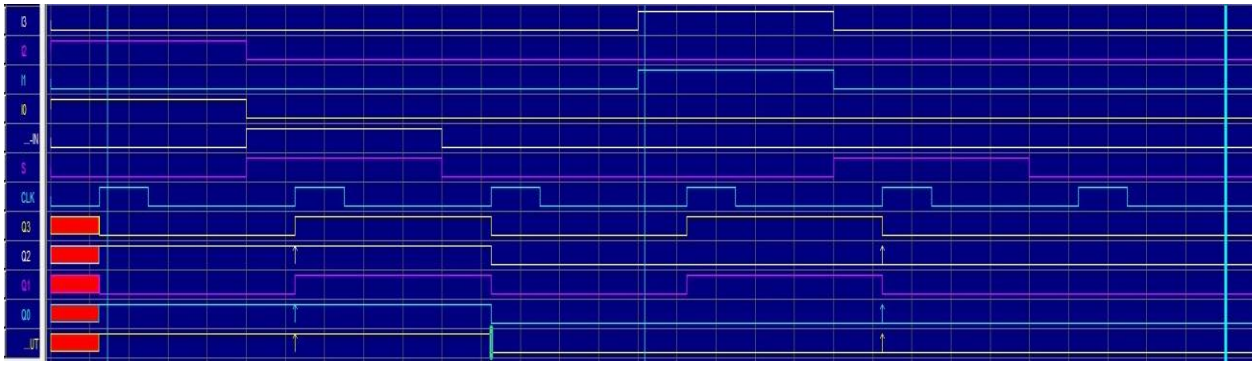
command file



c-shiftRegister - Notepad

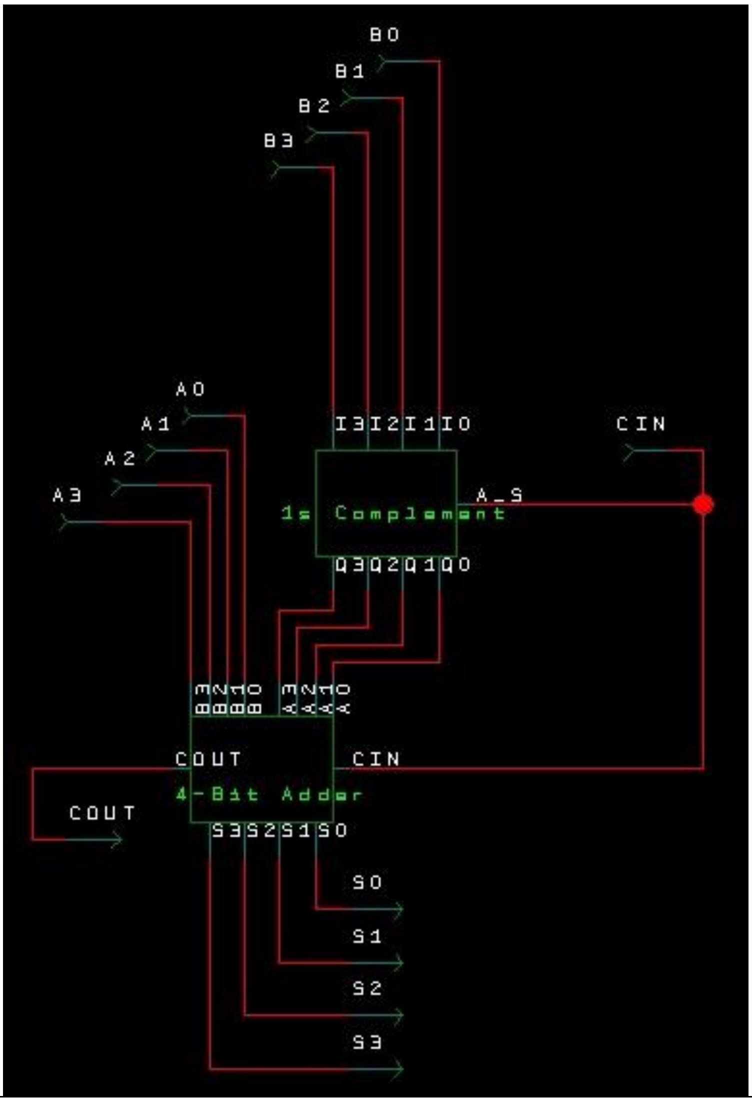
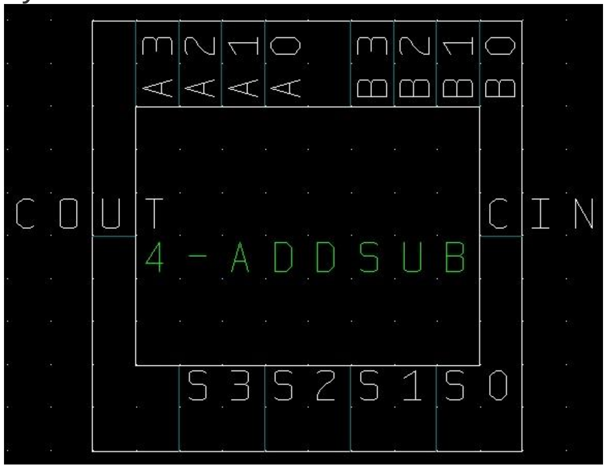
```
restart
wave c-shiftRegister.wfm I3 I2 I1 I0 SR-IN S CLK Q3 Q2 Q1 Q0 SR-OUT
stepsize 25.0ns
Clock CLK       0 1 0 0
pattern I3      0 0 0 1 0 0
pattern I2      1 0 0 0 0 0
pattern I1      0 0 0 1 0 0
pattern I0      1 0 0 0 0 0
pattern SR-IN   0 1 0 0 0 0
pattern S       0 1 0 0 1 0
sim 75ns
run
```

simulation result



Appendix section

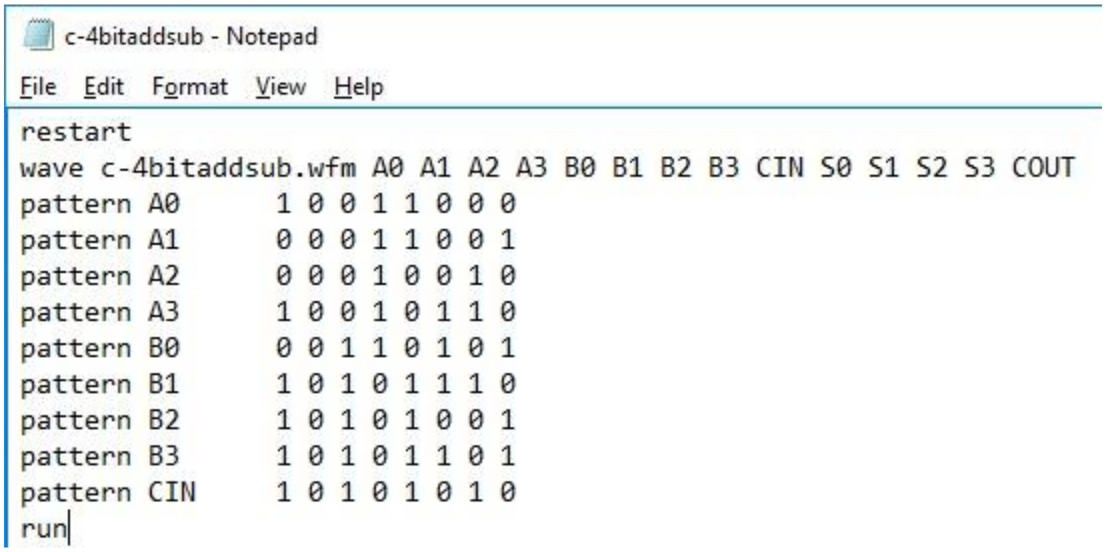
4-bit adder-Subtractor

Schematic	Symbol
 B0 B1 B2 B3 A0 A1 A2 A3 I3 I2 I1 I0 1s Complement A_S CIN Q3 Q2 Q1 Q0 B3 B2 B1 B0 A3 A2 A1 A0 COUT CIN 4-Bit Adder COUT S3 S2 S1 S0 S0 S1 S2 S3	 A3 A2 A1 A0 B3 B2 B1 B0 COUT CIN 4-ADDSUB S3 S2 S1 S0

expected results table

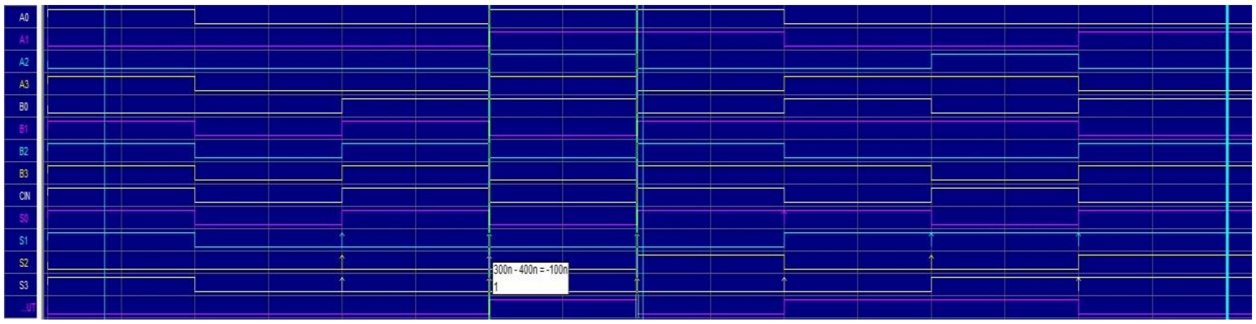
A0	B0	A1	B1	A2	B2	A3	B3	CIN	S0	S1	S2	S3	Cout
1	0	0	1	0	1	1	1	1	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	1	1	0	0	0	0
1	1	1	0	1	0	1	0	0	0	0	0	0	1
1	0	1	1	0	1	0	1	1	1	0	1	0	0
0	1	0	1	0	0	1	1	0	1	1	0	0	1
0	0	0	1	1	0	1	0	1	0	1	0	1	1
0	1	1	0	0	1	0	1	0	1	1	1	1	0

command file



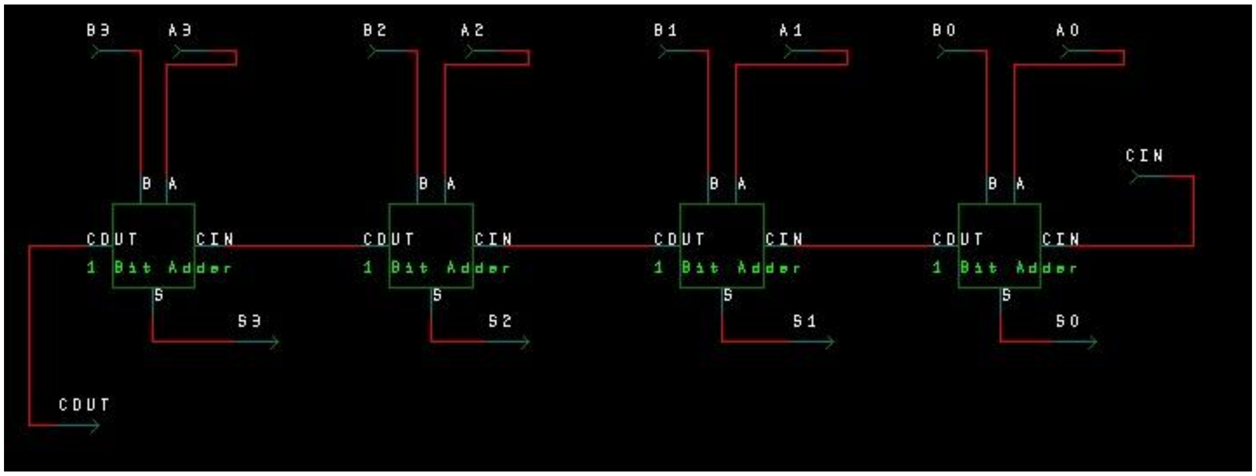
```
restart
wave c-4bitaddsub.wfm A0 A1 A2 A3 B0 B1 B2 B3 CIN S0 S1 S2 S3 COUT
pattern A0      1 0 0 1 1 0 0 0
pattern A1      0 0 0 1 1 0 0 1
pattern A2      0 0 0 1 0 0 1 0
pattern A3      1 0 0 1 0 1 1 0
pattern B0      0 0 1 1 0 1 0 1
pattern B1      1 0 1 0 1 1 1 0
pattern B2      1 0 1 0 1 0 0 1
pattern B3      1 0 1 0 1 1 0 1
pattern CIN     1 0 1 0 1 0 1 0
run
```

simulation result

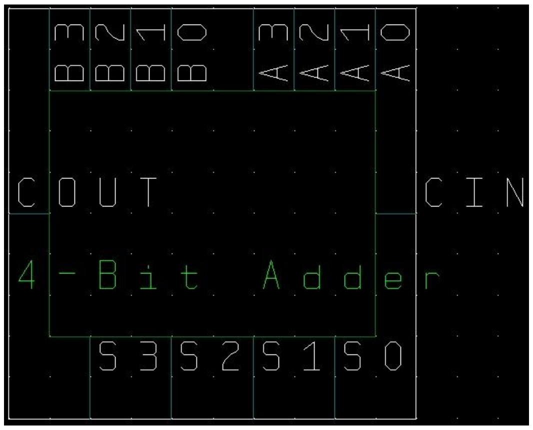


4-bit adder

Schematic



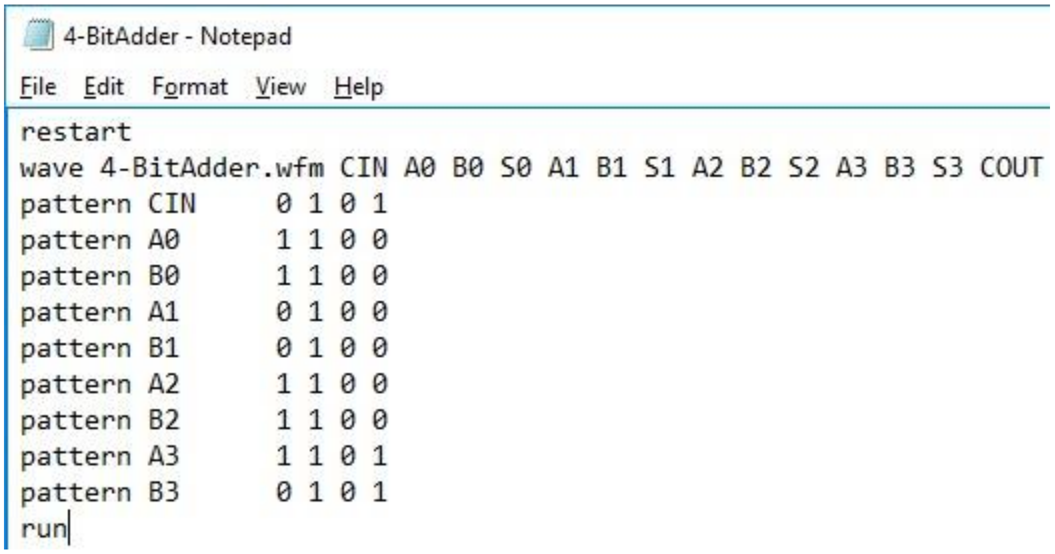
Symbol



expected results table

A0	B0	A1	B1	A2	B2	A3	B3	S0	S1	S2	S3	Cout
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	1	1	1	1	0
1	0	1	0	1	0	1	0	1	1	1	1	0
1	1	1	1	1	1	1	1	0	0	0	0	4

command file

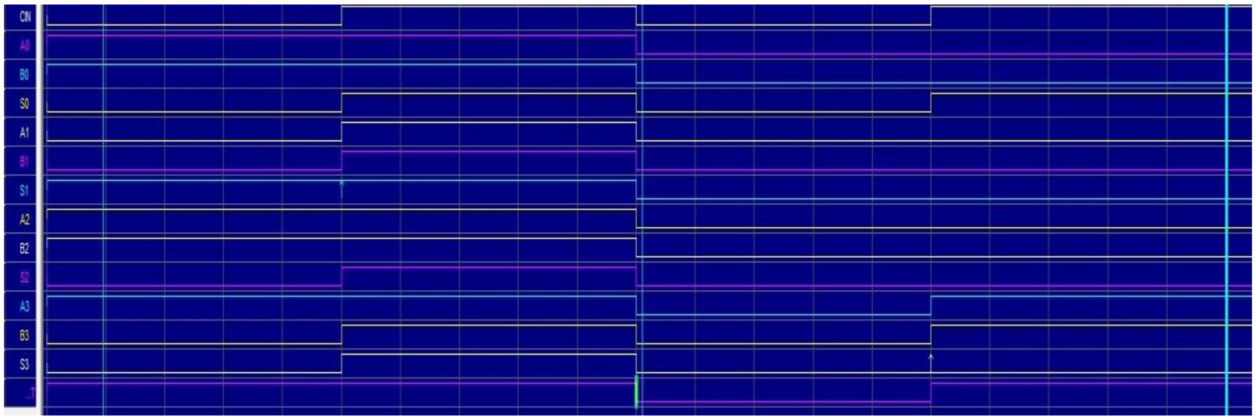


4-BitAdder - Notepad

File Edit Format View Help

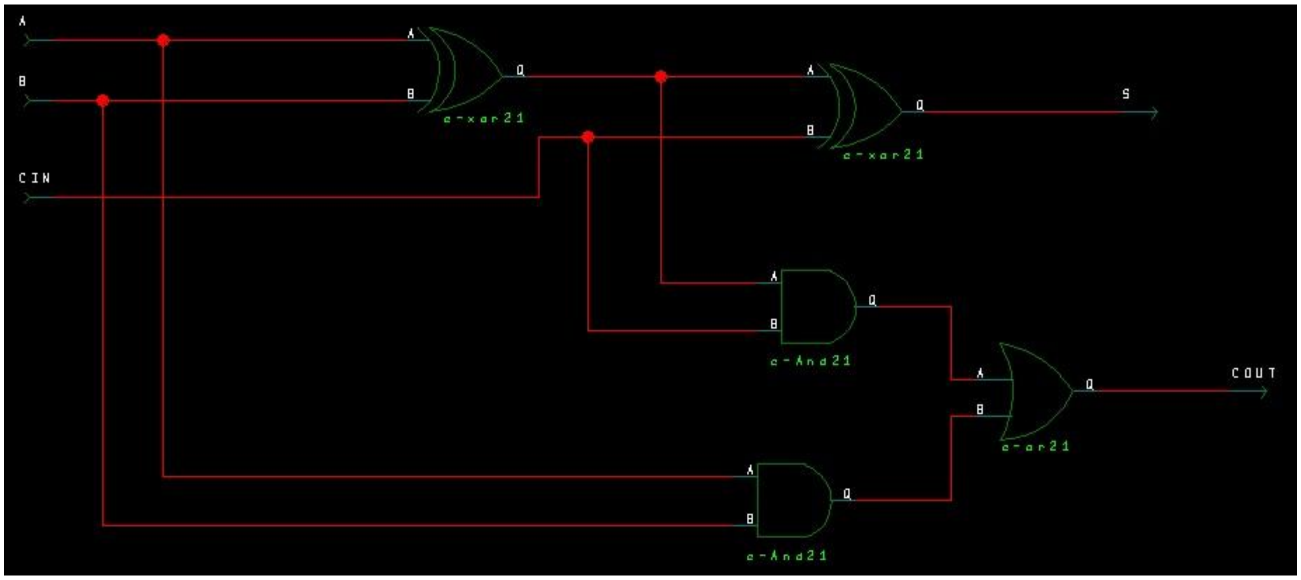
```
restart
wave 4-BitAdder.wfm CIN A0 B0 S0 A1 B1 S1 A2 B2 S2 A3 B3 S3 COUT
pattern CIN     0 1 0 1
pattern A0      1 1 0 0
pattern B0      1 1 0 0
pattern A1      0 1 0 0
pattern B1      0 1 0 0
pattern A2      1 1 0 0
pattern B2      1 1 0 0
pattern A3      1 1 0 1
pattern B3      0 1 0 1
run
```

simulation result

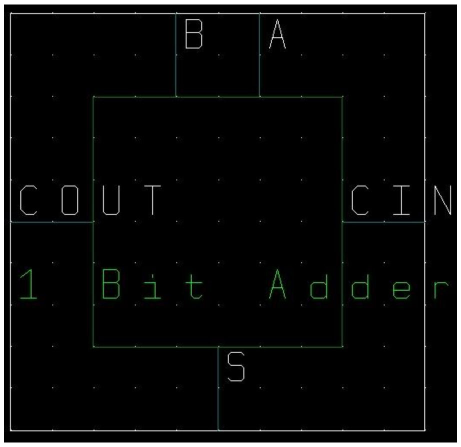


1-bit full adder

Schematic



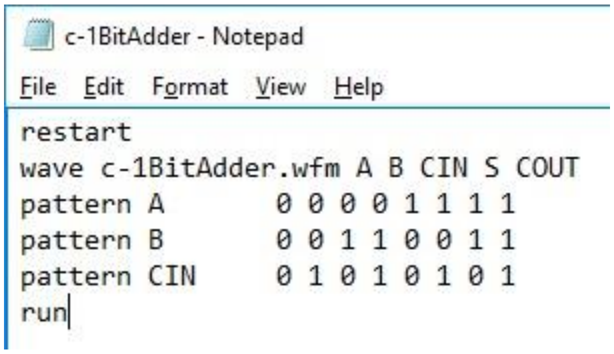
Symbol



expected results table

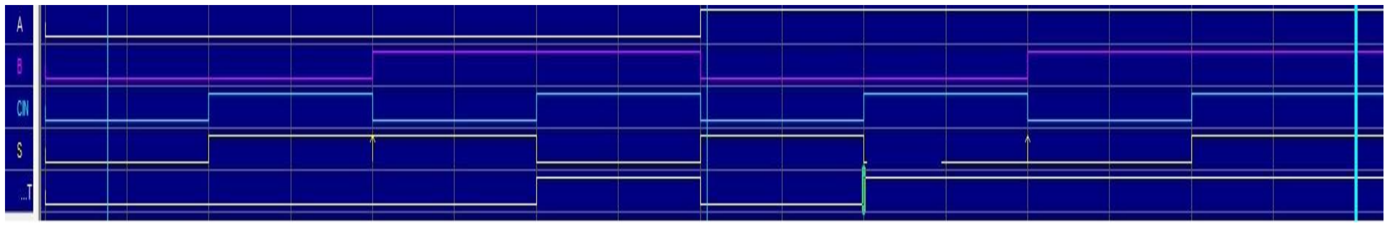
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

command file



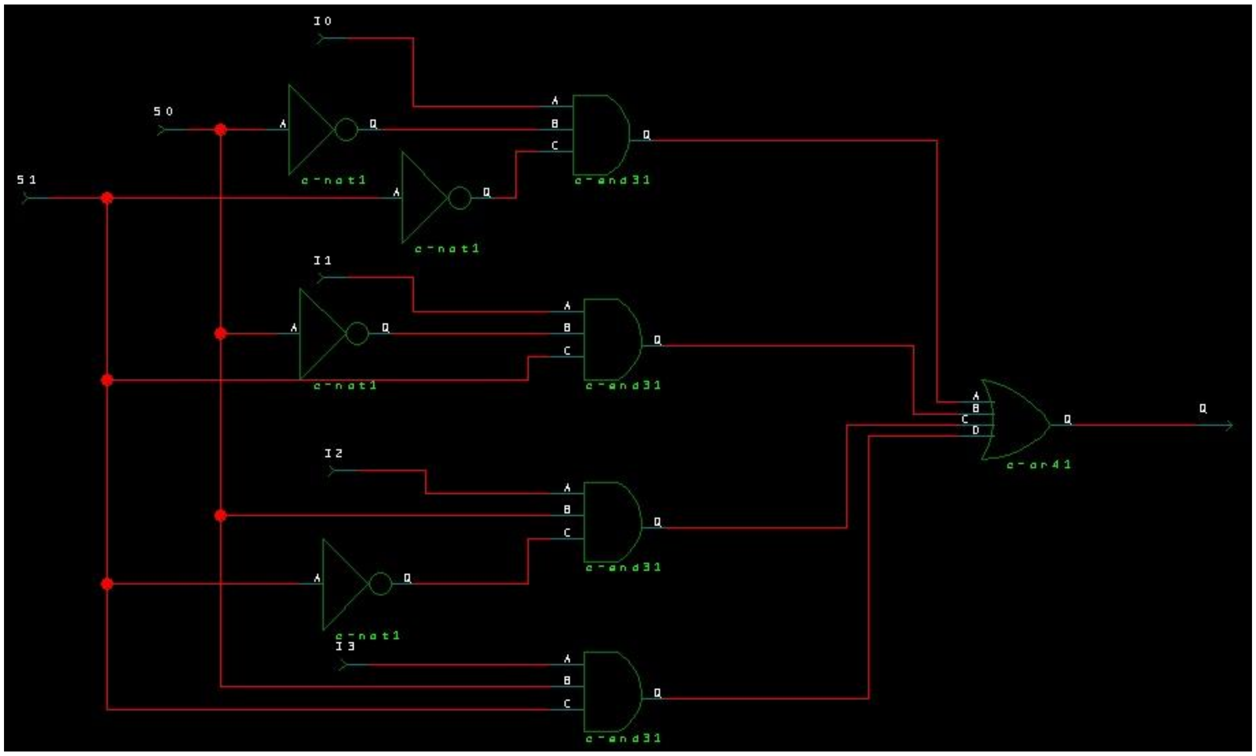
```
restart
wave c-1BitAdder.wfm A B CIN S COUT
pattern A       0 0 0 0 1 1 1 1
pattern B       0 0 1 1 0 0 1 1
pattern CIN     0 1 0 1 0 1 0 1
run
```

simulation result

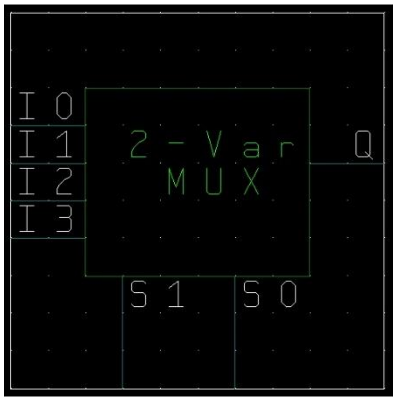


2-Var mux

Schematic



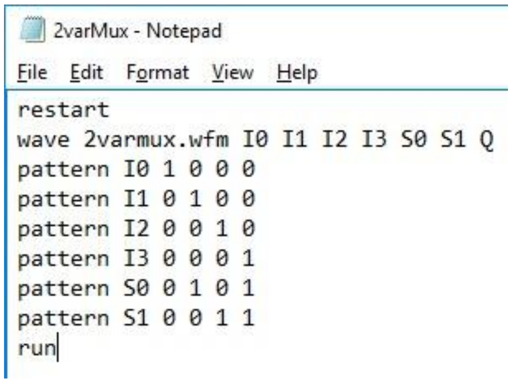
Symbol



expected results table

S0	S1	Q
0	0	I0
0	1	I1
1	0	I2
1	1	I3

command file

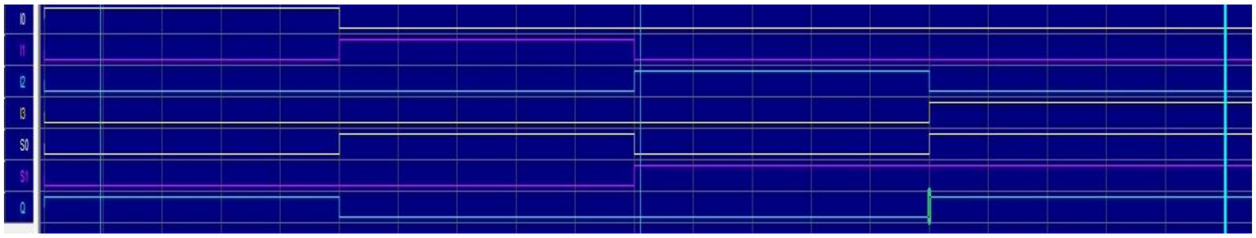


2varMux - Notepad

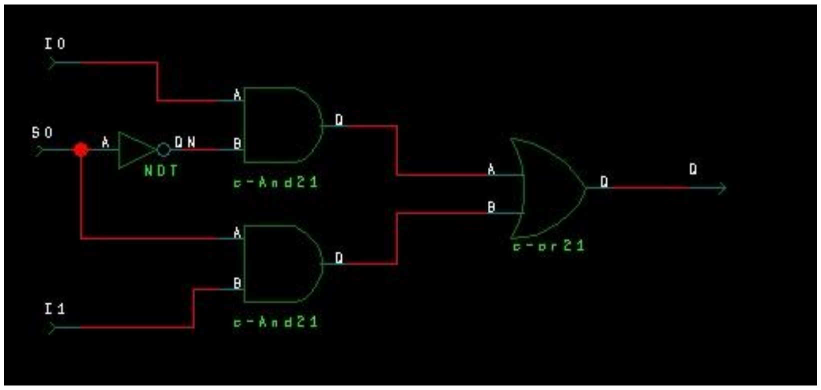
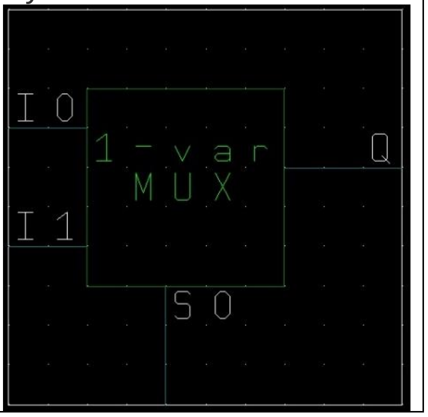
File Edit Format View Help

```
restart
wave 2varmux.wfm I0 I1 I2 I3 S0 S1 Q
pattern I0 1 0 0 0
pattern I1 0 1 0 0
pattern I2 0 0 1 0
pattern I3 0 0 0 1
pattern S0 0 1 0 1
pattern S1 0 0 1 1
run
```

simulation result



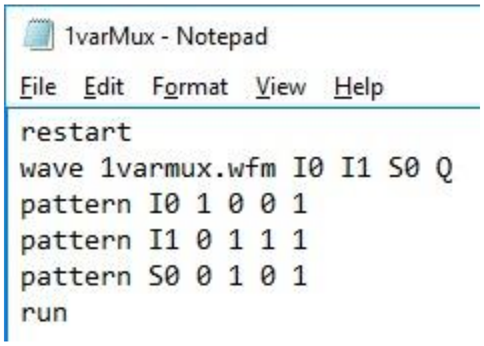
1-Var mux

Schematic	Symbol
 I0 S0 A QN NOT A B Q c-And21 A B Q c-And21 I1 A B Q c-or21 Q	

expected results table

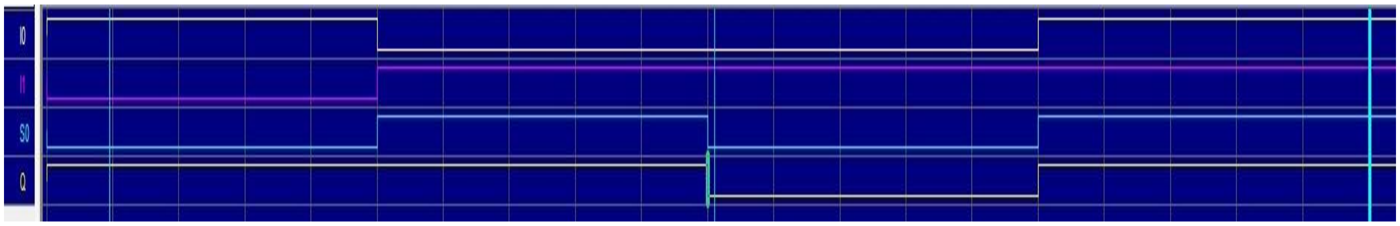
S0	I0	I1	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

command file

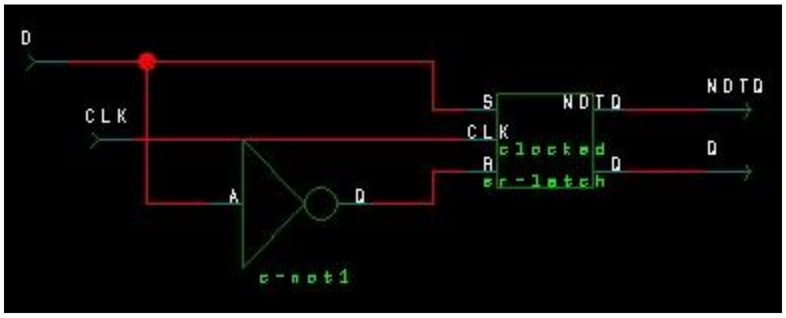
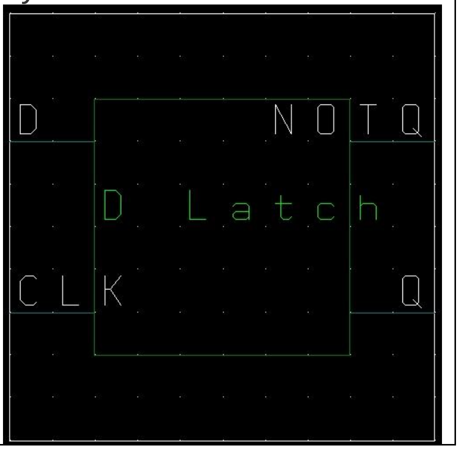


```
restart
wave 1varmux.wfm I0 I1 S0 Q
pattern I0 1 0 0 1
pattern I1 0 1 1 1
pattern S0 0 1 0 1
run
```

simulation result



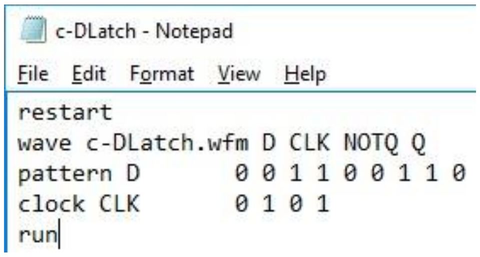
D-latch

Schematic	Symbol
 D CLK S NOTQ NOTQ CLK R clocked sr-latch Q Q A Q c-not1	

expected results table

CLK	D	Q	NOT Q
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

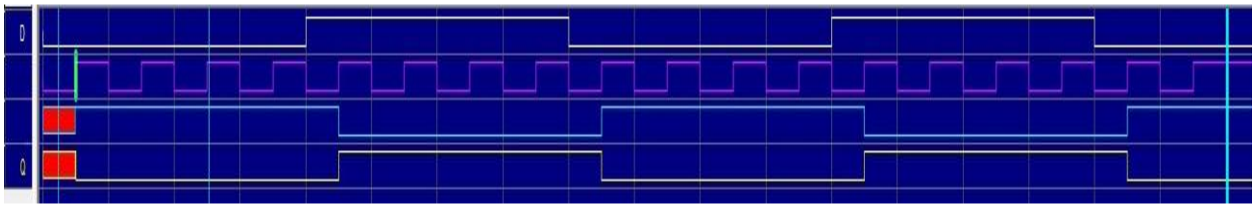
command file


c-DLatch - Notepad

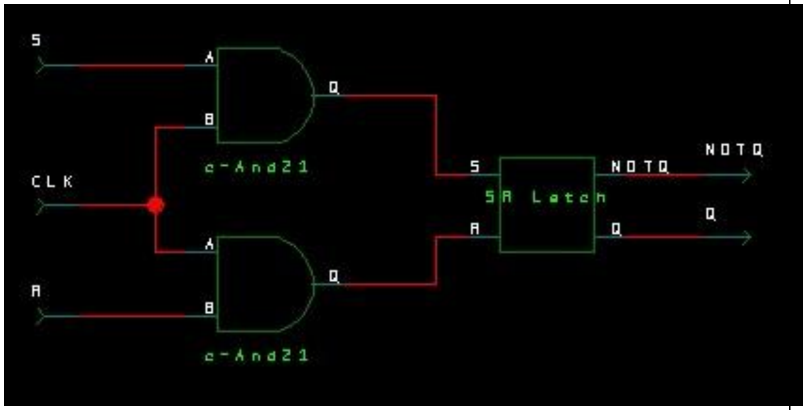
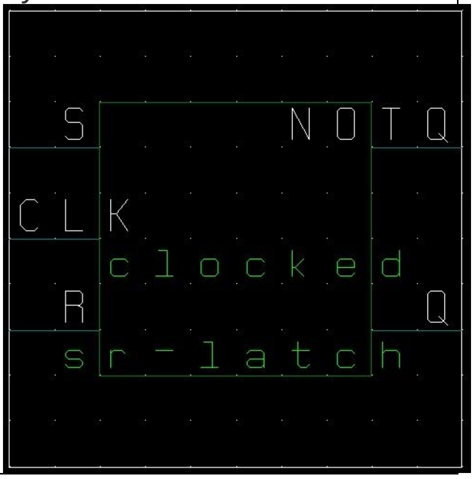
File Edit Format View Help

```
restart
wave c-DLatch.wfm D CLK NOTQ Q
pattern D       0 0 1 1 0 0 1 1 0
clock CLK       0 1 0 1
run
```

simulation result



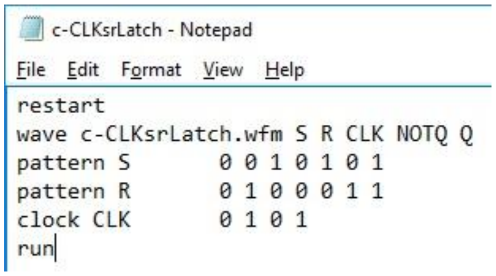
CLK sr-latch

Schematic	Symbol
	

expected results table

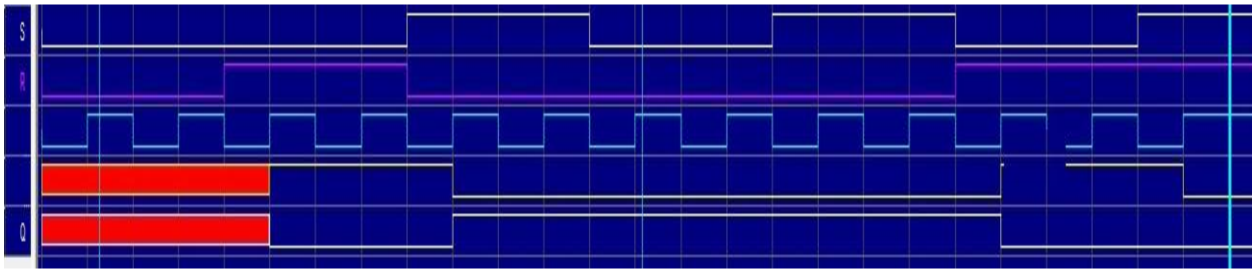
S	R	Q	NOT Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

command file

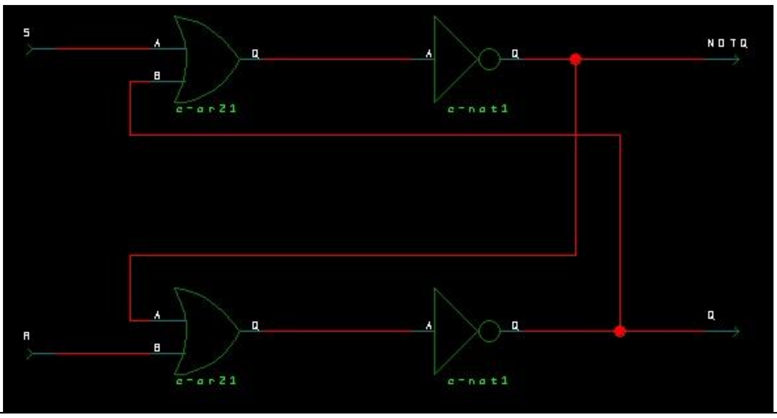
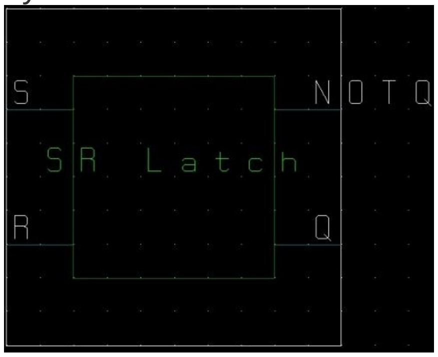


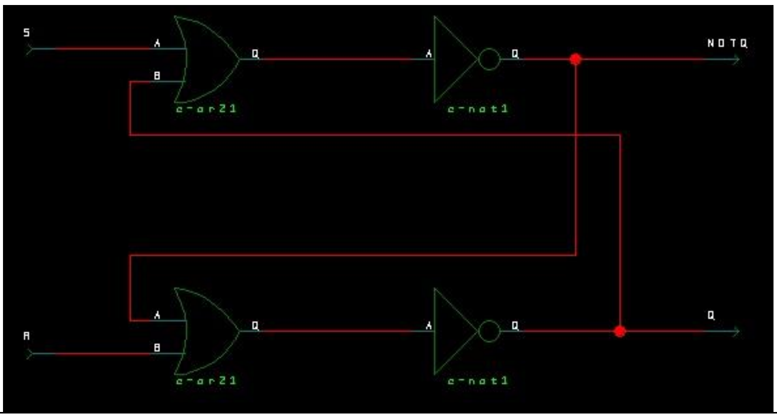
```
restart
wave c-CLKsrLatch.wfm S R CLK NOTQ Q
pattern S       0 0 1 0 1 0 1
pattern R       0 1 0 0 0 1 1
clock CLK       0 1 0 1
run
```

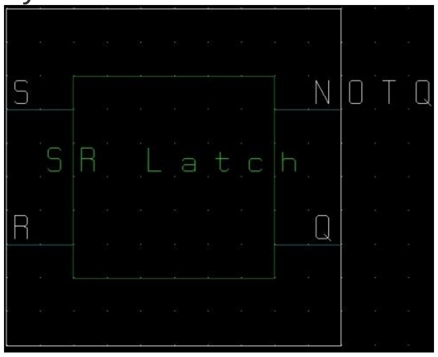
simulation result



sr-latch

Schematic	Symbol
	

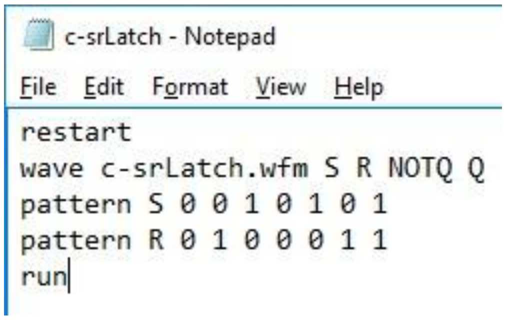




expected results table

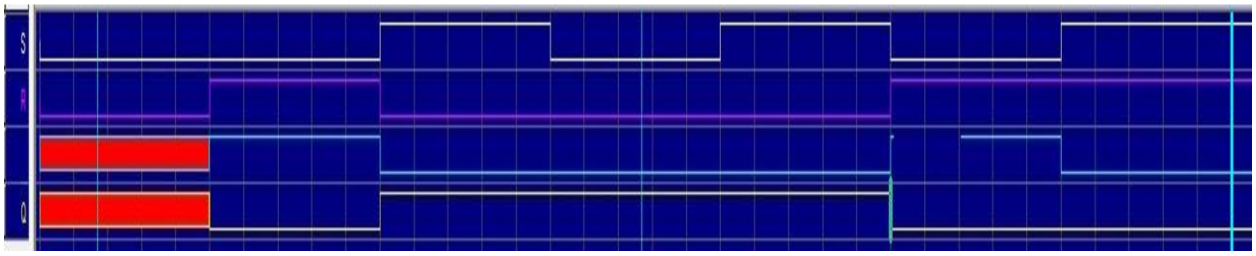
S	R	Q	NOT Q
0	0	LATCH	LATCH
0	1	0	1
1	0	1	0
1	1	0	0

command file

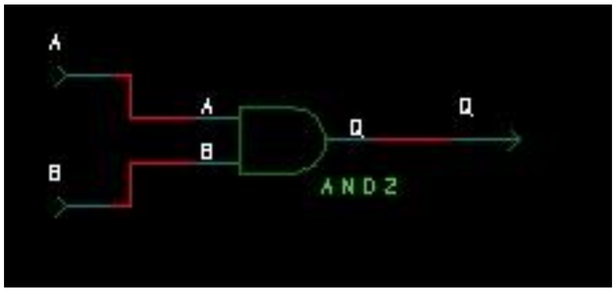
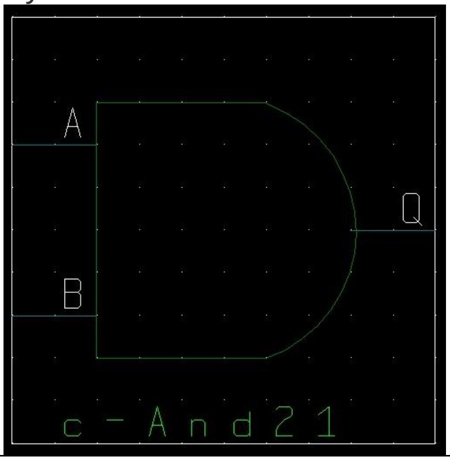


```
restart
wave c-srLatch.wfm S R NOTQ Q
pattern S 0 0 1 0 1 0 1
pattern R 0 1 0 0 0 1 1
run
```

simulation result



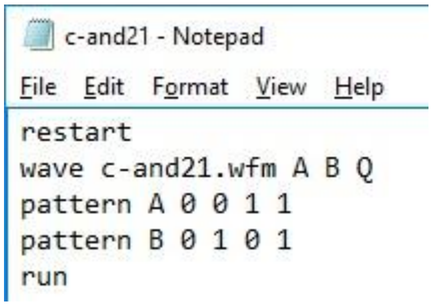
c-and21

Schematic	Symbol
	

expected results table

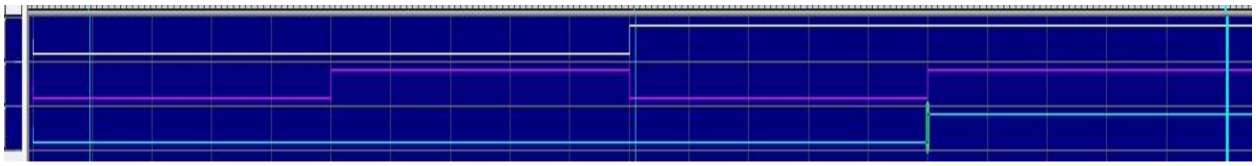
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

command file

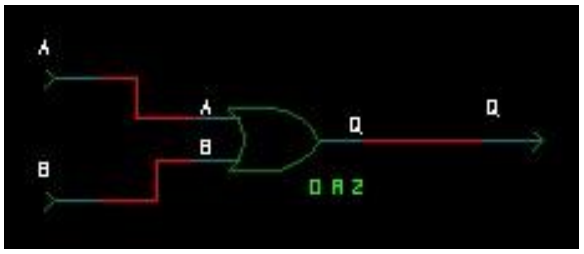
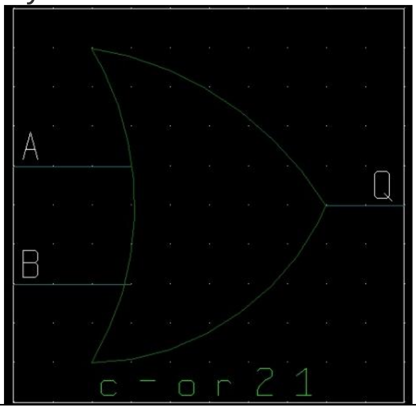


```
restart
wave c-and21.wfm A B Q
pattern A 0 0 1 1
pattern B 0 1 0 1
run
```

simulation result



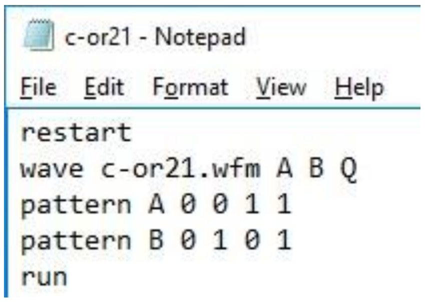
c-or21

Schematic	Symbol
	

expected results table

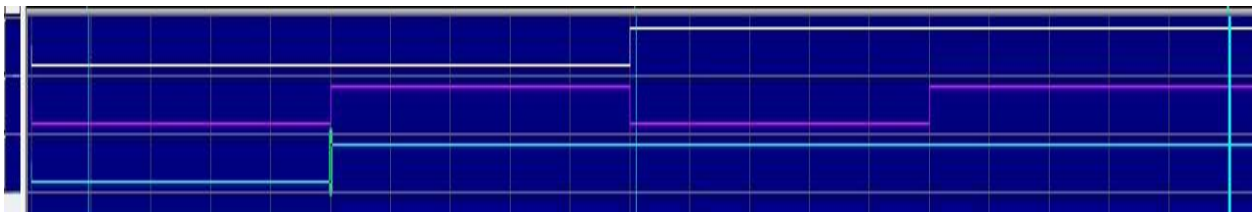
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

command file

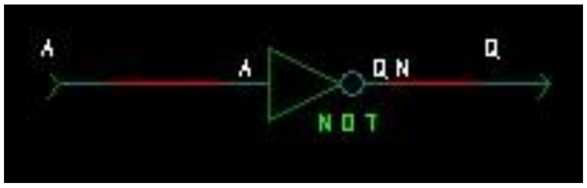
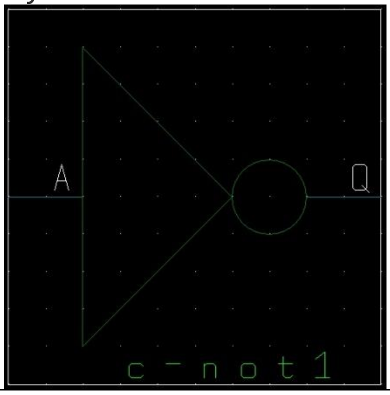


```
restart
wave c-or21.wfm A B Q
pattern A 0 0 1 1
pattern B 0 1 0 1
run
```

simulation result



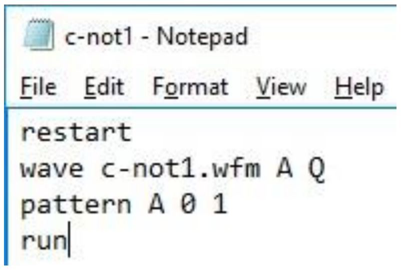
c-not1

Schematic	Symbol
	

expected results table

A	Q
0	1
1	0

command file



c-not1 - Notepad

File Edit Format View Help

```
restart
wave c-not1.wfm A Q
pattern A 0 1
run
```

simulation result