Lab Report

By Justin Le and Chris Daly

7/21/16

Logic and Computer Systems

OBJECTIVE AND ANALYSIS

In the beginning of our labs, we spent time designing our own custom schematics, starting from the simplest components and building upward from there. We began by designing the most basic components, such as c-and21, c-or21, c-not1, and etc. From there, we used these basic components to build the more complex components, such as the exclusive or. These components continuously stacked in order to create more complex structures, which were in the end chained together to create an ALU.

However, it was not completely an easy task to complete. During this project, we encountered many problems that we had to overcome. One such problem was that even though we had labs that looked right, when stacked to create a more complex schematic, errors were found at a later date. These errors were frustratingly difficult to locate, later in the lab. For example, we had difficulty understanding the functionality of the shift register, which resulted in us making modifications. However, these modifications were incorrect, and later, we had to logically follow the exact flow of information to correct our mistakes. Some other difficulties were found when we thought we had adequate test data but made minor mistakes with the mathematics involved. To summarize, without a strong logical understanding of some of the material, it was often difficult to figure out where exactly we went wrong.

One of the most frustrating parts of this experience was the software we were using. The software is super cluttered and often very finicky about what it will and will not allow. Often naming of variables would obtain a typo and this would also throw our whole schematic in disarray. However, it was also very difficult to locate these errors due to vague error messages. We could list all of the individual issues that we experienced while manipulating dash, but we run out of room for the rest of our objective and analysis, and therefore will end this portion here.

In order to prepare for these classes, we took detailed notes during all of the class lectures. These notes were examined on our own during our time at home and after class. During times when understanding of the material was poor, we would often refer to outside sources or the textbook for additional information. Throughout the project, we also learned about team work and efficiency. Since the material was often a big task, and there was always a deadline approaching, we were forced to be efficient in order to finish our tasks in the proper amount of time.

We both learned a few things during our experiences throughout this project. We practiced logical gates and how they fit into the larger schematics. We took these logical gates and learned circuit board designs of various difficulties. We also practiced preparation tactics for tackling a larger project. Overall, these lab projects were a great learning experience for both of us, and it strengthened our understanding of the material.

Detailed Technical Explanation

In our lab, our objective was to create and implement a functional ALU that is built of components that we created in our previous labs. The components will be used to implement a four-bit shift register, which will take in four bits, record them in memory, and shift them as necessary. In this detailed technical explanation, we will discuss these individual components. We began by creating a list of all the tasks that the ALU aimed to complete. These were the Adder/subtractor, AND, pass, inverter, and ALU outputter (whose function is to determine which of these functions to output). So, each function was given it's on module, which made the design of the ALU simpler when designing as a whole. This allowed the ALU shift register to be implemented in a much simpler fashion than we originally believed.

First of all, we would to discuss the adder/subtractor component of our ALU. The adder /subtractor takes in two four-bit binary numbers and a carry in variable. The adder/subtractor is also made up of two components—a one's compliment for use of subtraction and a 4-bit adder which handles the addition. The one's compliment will only trigger if the Carry in variable is one. Otherwise, the one's compliment will have no effect on the overall value of the four-bit number. The "Adder" takes in two 4-bit numbers and adds them together using binary logic. Simply speaking, the four-bit is made up of four 1-bit adders. The one-bit adders simply take in two one bit numbers and add them together and calculate the carry based on binary addition logic.

The second component we would like to talk to you about is a "4-bit and gate". The purpose of this component is to calculate the "and" values for each of the two 4-bit numbers. For example, A0 "AND" B0 are outputted to Q0. This is an extremely simple process, however in order to not cause clutter on our ALU, this is its own component.

Thirdly, the ALU needs a method for simply passing values through, in the case that there is no manipulation of the 4-bit number. This does not require its own module because literally nothing is being done to the values, and they can just be passed through to ALU outputter.

Fourthly, the inverter. The inverter is another very simple module for the ALU. It takes the values of the 4-bit number and runs them through an inverter. The inverters function is simple and all it does is take in "A" and output "not A". This was also put into its own component to avoid screen clutter on the ALU schematic.

Fifth, we will discuss our ALU outputter. This is the most complex part of our ALU schematic, and is actually the purpose that the ALU exists. Because the ALU's primary job is to effectively select between the jobs: Add, subtract, "AND", pass, or

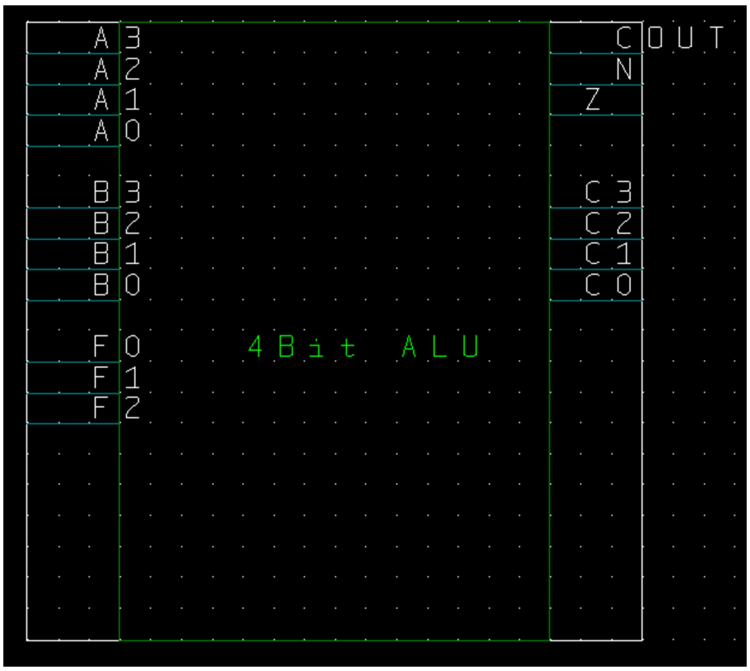
invert, given a 4-bit binary input. Since this was such an important component, it also was put into its own module. This module takes in two 4-bit binary numbers, as well as two selectors, F1 and F2. The selectors are used in order to determine what the final output of the ALU will be, between the calculations of add, subtract, “AND”, pass, or invert. It does this by implementing four two variable multiplexors. Each of these multiplexors accepts the values for a given binary digit. For example, the variable multiplexor that outputs to Q0 will take in values from ADD0, AND0, PASS0, and INV0, depending on the value of the selector variables. The selectors indicate add when they are 0 0, respectively. If the selectors are 0 1, this indicates an AND will be performed. At 1 0, the selectors will simply pass the values contained within the inputted 4-bit variable. Finally, at 1 1, the selectors will invert the inputted 4-bit variable.

Finally, the ALU needs to output two variables, N and Z. These variables function like follows: If N is 1, the output is a negative number; If Z is 1, the output will be 0. If the output is a positive non-zero number, N will be 0, and Z will be 0. These variables are calculated by reading the output of the ALU output, labeled C0 through C3. In conclusion, through all of these schematics, designs, symbols, truth tables, and lob projects, we have gained a better understanding of how Logic and Computing Systems functions.

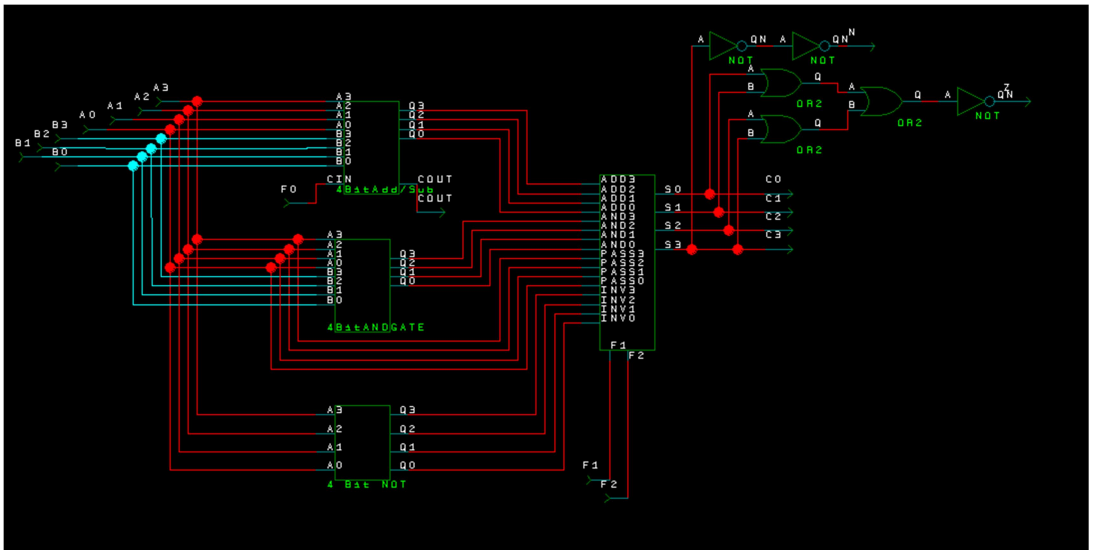
Symbol and Schematic Design

ALU

Symbol



Schematic



Command File

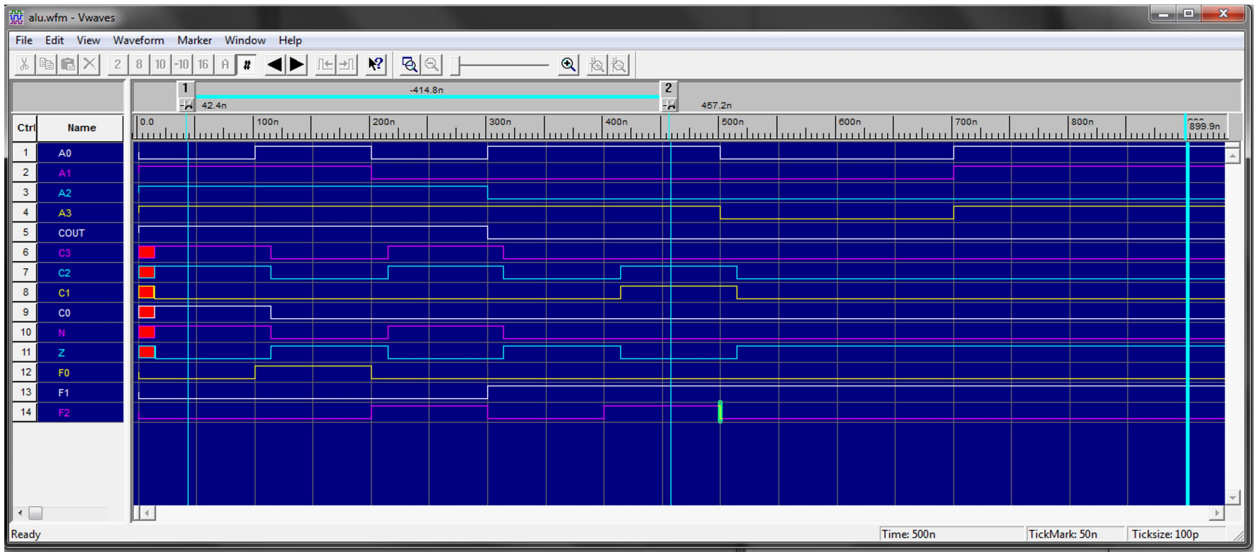
```
restart
wave alu.wfm A0 A1 A2 A3 COUT CLK SR LIN C3 C2 C1 C0 N Z F0 F1 F2 OUT
pattern A0 0 1 0 1 1 0 0 1 1
pattern A1 1 1 0 0 0 0 0 1 1
pattern A2 1 1 1 0 0 0 0 0 0
pattern A3 1 1 1 1 1 0 0 1 1
pattern B0 1 1 1 0 0 0 0 0 0
pattern B1 1 1 0 1 1 0 0 0 0
pattern B2 1 1 1 1 1 0 0 0 0
pattern B3 1 1 0 0 0 0 0 0 0
pattern F0 0 1 0 0 0 0 0 0 0
pattern F1 0 0 0 1 1 1 1 1 1
pattern F2 0 0 1 0 1 0 0 0 0
pattern SR 0 0 0 0 0 0 1 0 1
pattern LIN 0 0 0 0 1 0 0 0
pattern CLK 0 1 0 1 0 1 0 1
run
```

Expected Results

alu

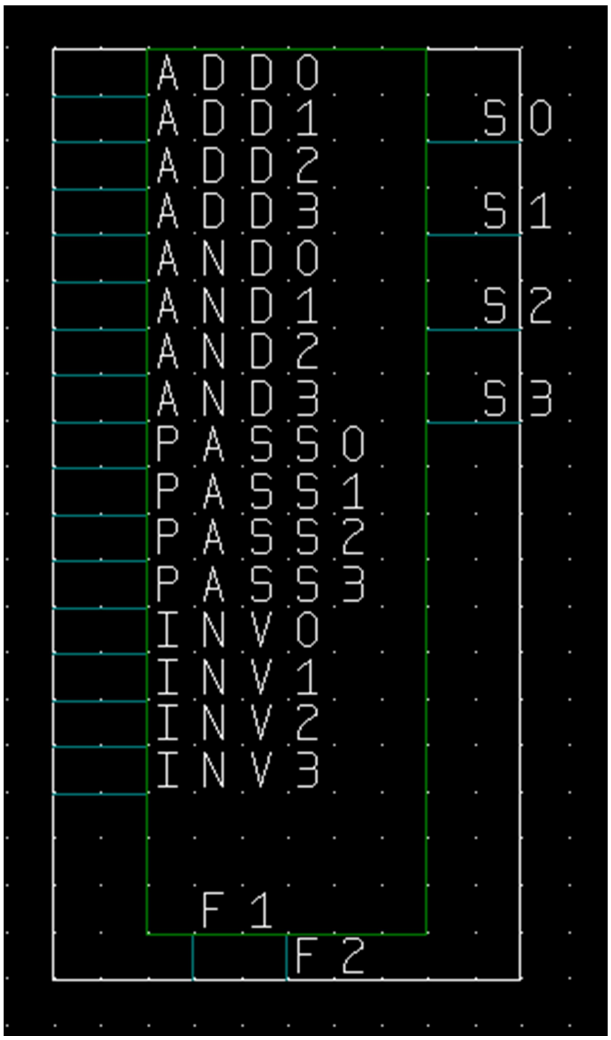
A3	A2	A1	A0	B3	B2	B1	B0	C3	C2	C1	C0	F0	F1	F2	Cout	N	Z
1	1	1	0	1	1	1	1	1	1	0	1	0	0	0	1	0	0
1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1
1	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	0	1	0	0	1	0	1	0	0	0	0
1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	0	0

Simulation

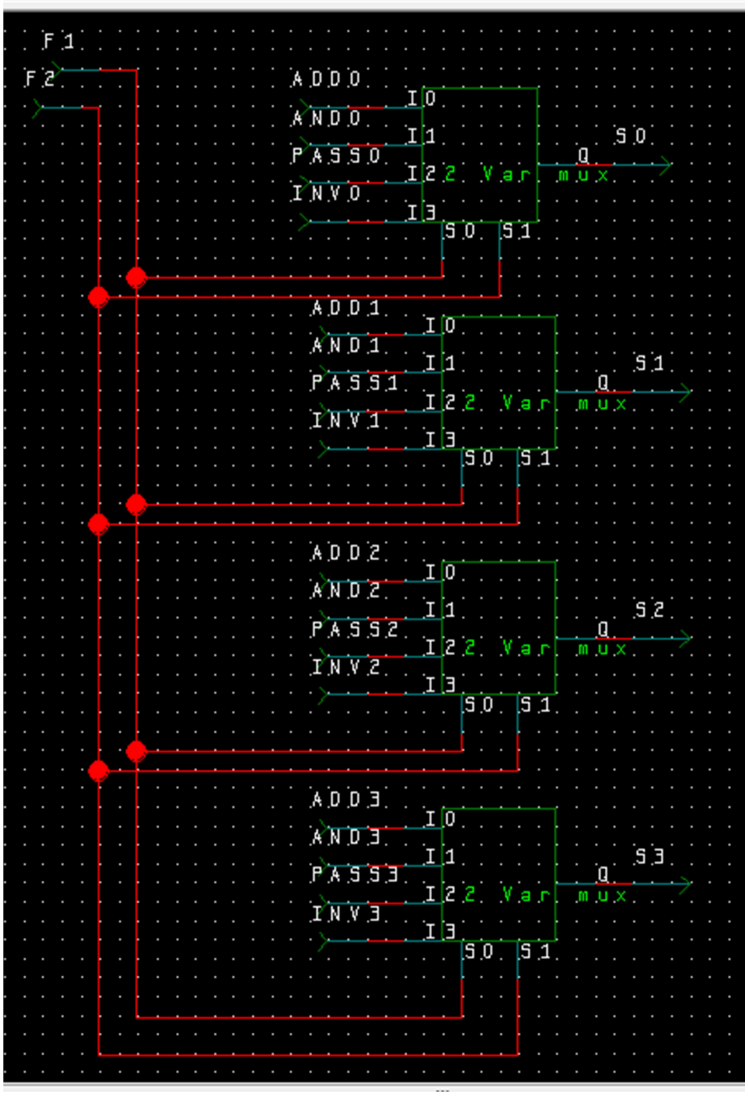


ALU OUTPUTTER

Symbol



Schematic


F1
F2
ADD0
AND0
PASS0
INV0
I0
I1
I2
I3
2 Var mux
O
S0
S0
S1
ADD1
AND1
PASS1
INV1
S1
ADD2
AND2
PASS2
INV2
S2
ADD3
AND3
PASS3
INV3
S3

Command File

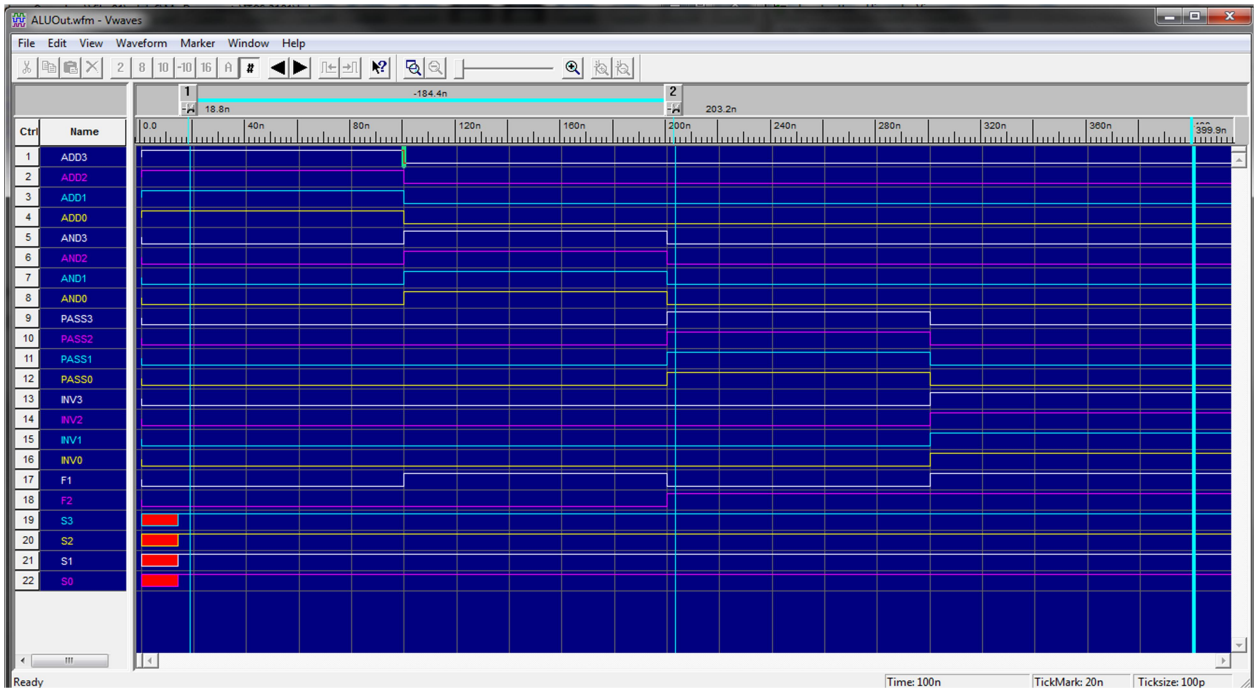
```
restart
wave ALUOut.wfm ADD3 ADD2 ADD1 ADD0 AND3 AND2 AND1 AND0 PASS3 PASS2 PASS1 PASS0 INV3
INV2 INV1 INV0 F1 F2 S3 S2 S1 S0
pattern ADD3 1 0 0 0
pattern ADD2 1 0 0 0
pattern ADD1  1 0 0 0
pattern ADD0 1 0 0 0
pattern AND3 0 1 0 0
pattern AND2 0 1 0 0
pattern AND1 0 1 0 0
pattern AND0 0 1 0 0
pattern PASS3 0 0 1 0
pattern PASS2 0 0 1 0
pattern PASS1  0 0 1 0
pattern PASS0 0 0 1 0
pattern INV3 0 0 0 1
pattern INV2 0 0 0 1
pattern INV1  0 0 0 1
pattern INV0 0 0 0 1
pattern F1 0 1  0 1
pattern F2 0 0 1 1
run
```

Expected Results

ALU outputter

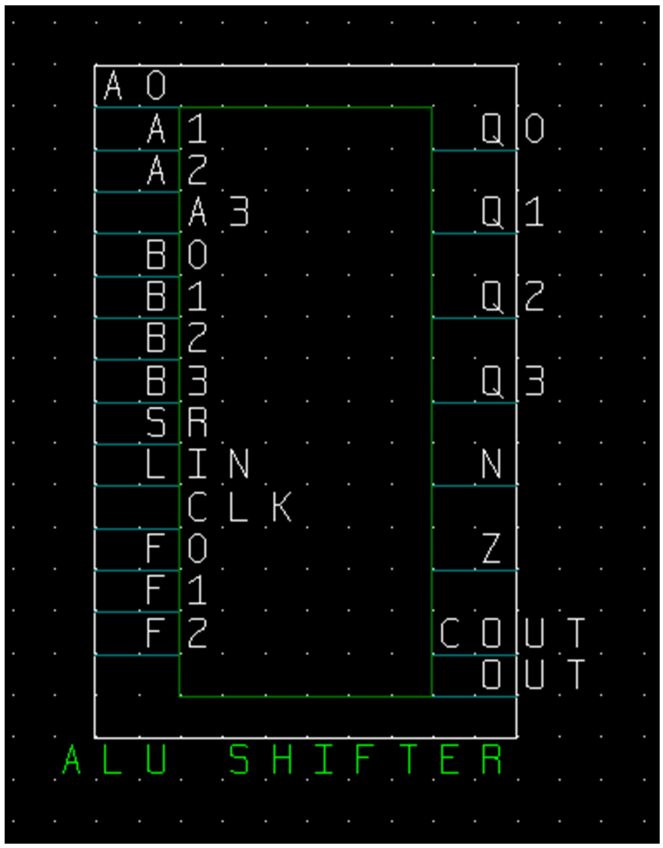
ADD3	ADD2	ADD1	ADD0	AND3	AND2	AND1	AND0	PASS3	PASS2	PASS1	PASS0	INV3	INV2	INV1	INV0	F1	F2	S3	S2	S1	S0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Simulation

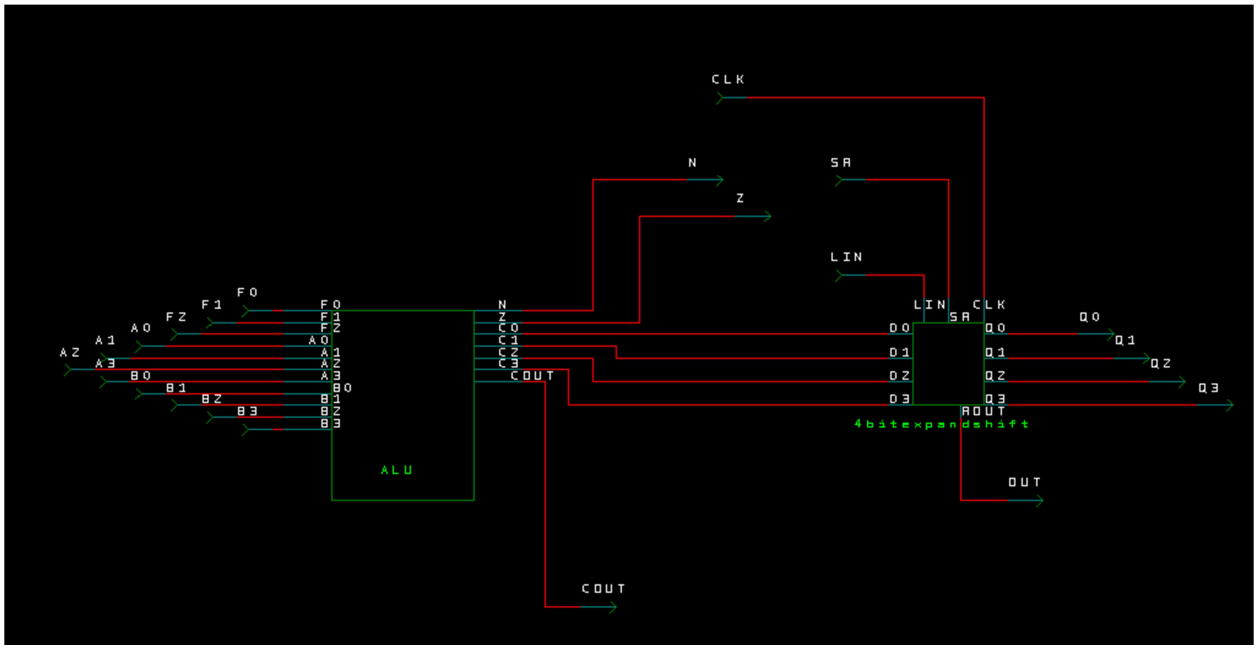


ALU SHIFTER

Symbol



Schematic



Command File

restart

wave alu.wfm A0 A1 A2 A3 COUT CLK SR LIN Q3 Q2 Q1 Q0 N Z F0 F1 F2 OUT

pattern A0 0 1 0 1 1 0 0 1 1

pattern A1 1 1 0 0 0 0 0 1 1

pattern A2 1 1 1 0 0 0 0 0 0

pattern A3 1 1 1 1 1 0 0 1 1

pattern B0 1 1 1 0 0 0 0 0 0

pattern B1 1 1 0 1 1 0 0 0 0

pattern B2 1 1 1 1 1 0 0 0 0

pattern B3 1 1 0 0 0 0 0 0 0

pattern F0 0 1 0 0 0 0 0 0 0

pattern F1 0 0 0 1 1 1 1 1 1

pattern F2 0 0 1 0 1 0 0 0 0

pattern SR 0 0 0 0 0 0 1 0 1

pattern LIN 0 0 0 0 1 0 0 0

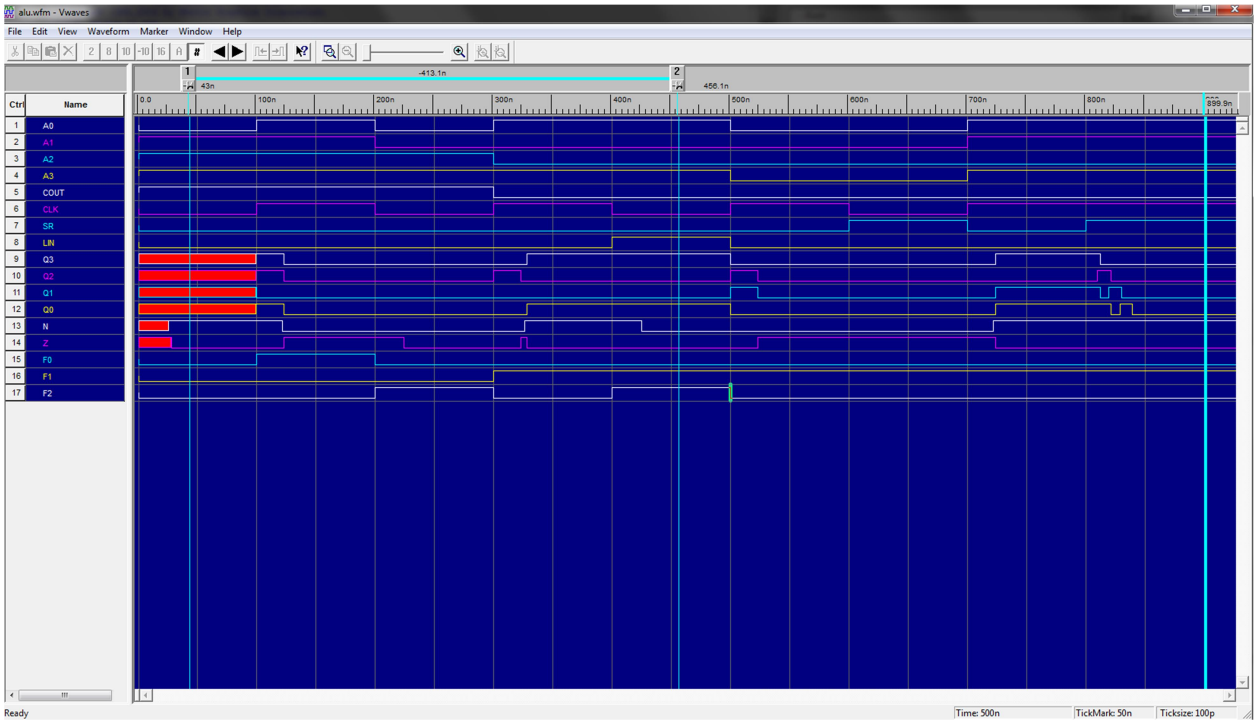
pattern CLK 0 1 0 1 0 1 0 1

run

Expected Results

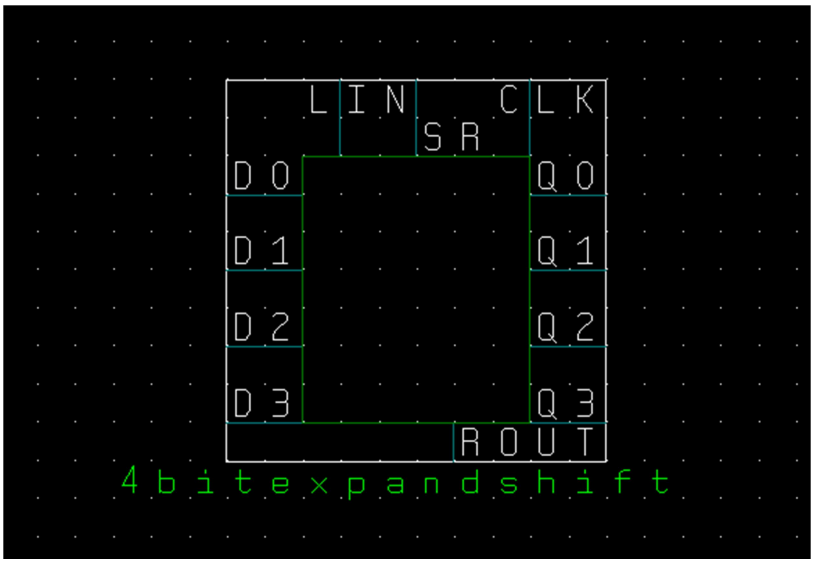
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	A3	A2	A1	A0	#	B3	B2	B1	B0	#	F	shift	Value	OUTPUT		Carry
2	1	1	1	0	14	1	1	1	1	15	+	no	0	1101		1
3	1	1	1	1	15	1	1	1	1	15	-	no	0	0000		0
4	1	1	0	0	12	0	1	0	1	5	And	no	0	0010		0
5	1	0	0	1	9	0	1	1	0	6	A	no	0	1001		0
6	1	0	0	1	9	0	1	1	0	0	NotA	no	0	0110		0
7	0	0	0	0	0	0	0	0	0	0	A	no	0	0000		0
8	0	0	0	0	0	0	0	0	0	0	A	yes	1	1000		0
9	1	0	1	1	11	0	0	0	0	0	A	no	0	1011		0
10	1	0	1	1	11	0	0	0	0	0	A	yes	1	1101		0

Simulation

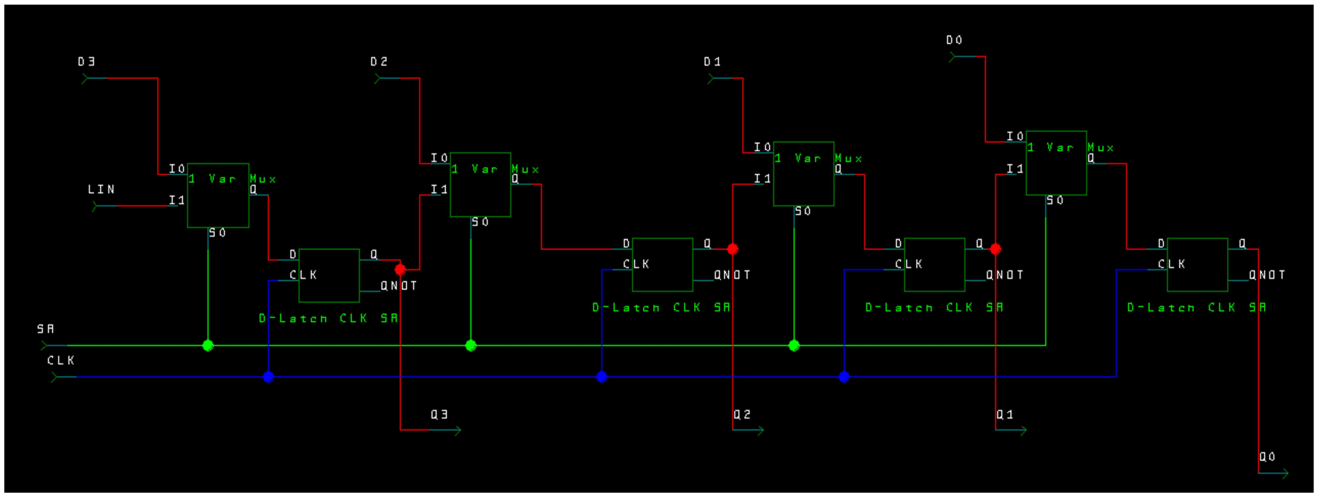


4-bit Shift Register

Symbol



Schematic



Command File

restart

wave 4-bit-expandable-shift-register.wfm SR CLk Lin D3 D2 D1 D0 Q3 Q2 Q1 Q0

pattern SR 0 1 0 1

pattern CLk 0 1 0 1

pattern Lin 0 0 0 1

pattern D3 0 1 0 0

pattern D2 0 0 0 1

pattern D1 0 1 0 0

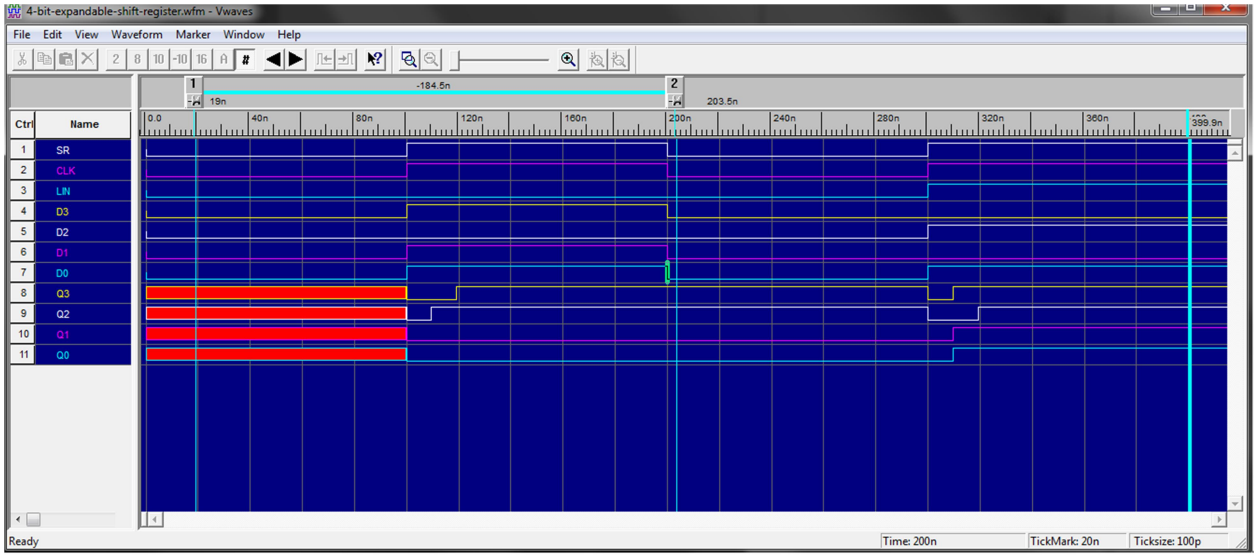
pattern D0 0 1 0 1

run

Expected Results

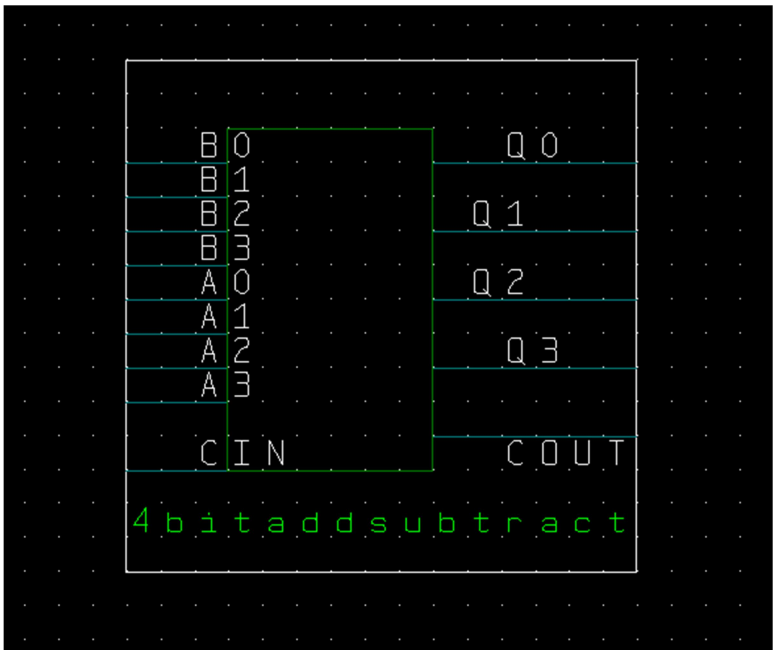
SR	CLK	LIN	D3	D2	D1	D0		Q3	Q2	Q1	Q0	4 bit shift reg
0	0	0	0	0	0	0		0	0	0	0	
1	1	0	1	0	1	1		1	1	0	0	
0	0	0	0	0	0	0		0	0	0	0	
1	1	1	0	1	0	1		1	1	1	1	

Simulation

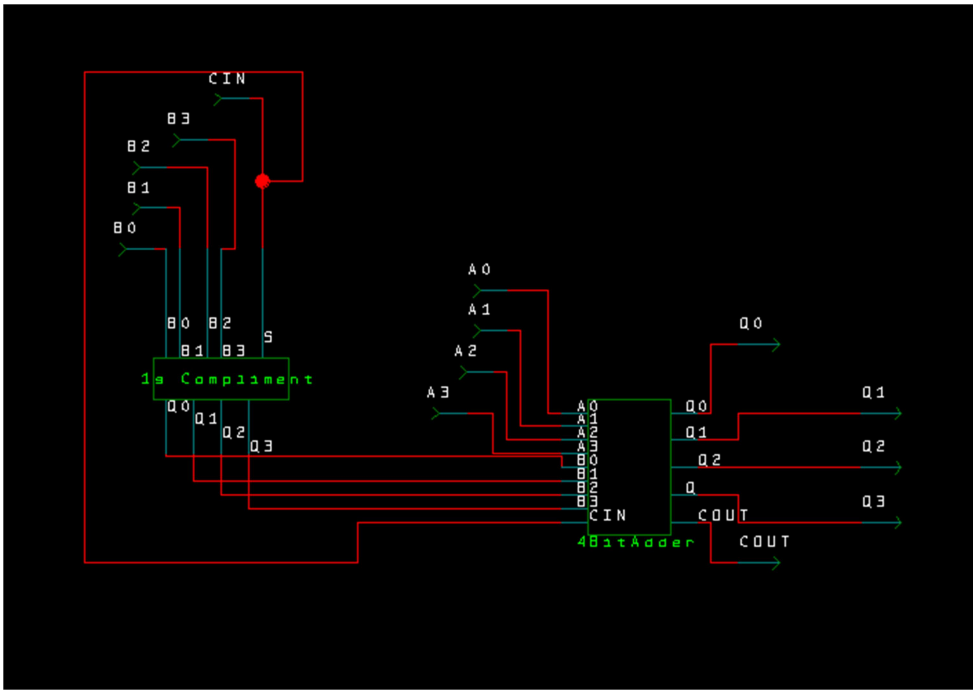


4-bit Adder/Subtractor

Symbol



Schematic



Command File

restart

wave 4bitAdderSubtracter.wfm A0 B0 CIN COUT SUM Q0

pattern A0 0 0 1 1

pattern B0 0 1 0 1

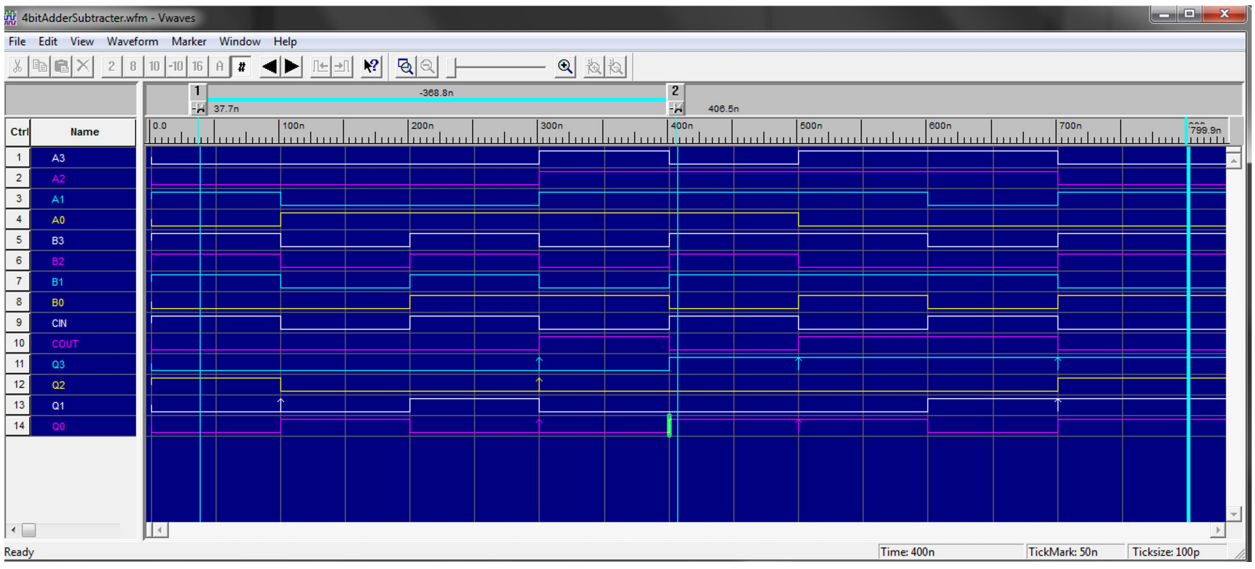
pattern CIN 1 0 1 0

run

Expected Results

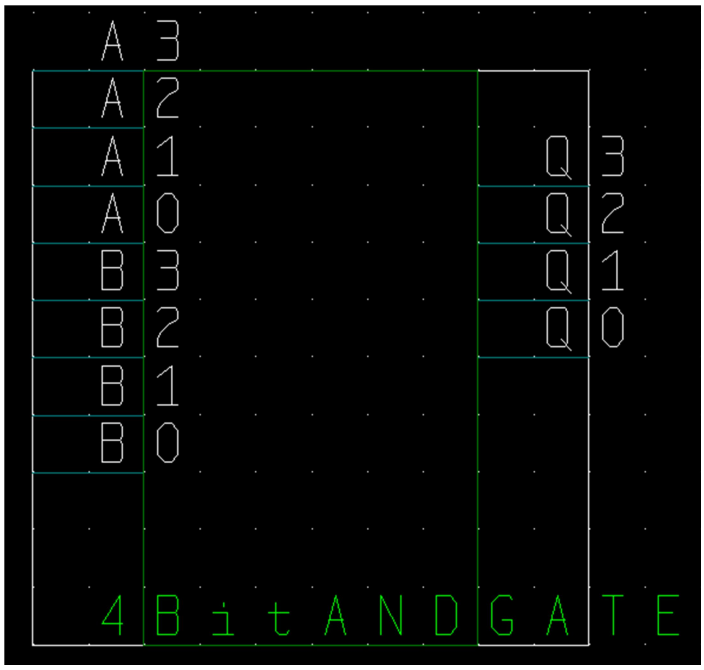
A3	A2	A1	A0	B3	B2	B1	B0	Cin	Cout	Q3	Q2	Q1	Q0	4 bit adder sub
0	0	1	0	1	1	1	0	1	0	0	1	0	0	
0	0	0	1	0	0	0	0	0	0	0	0	0	1	
0	0	0	1	1	1	1	1	1	0	0	0	1	0	
1	1	1	1	0	0	0	1	0	1	0	0	0	0	
0	1	1	1	1	1	1	0	1	0	1	0	0	1	
1	1	1	0	1	0	1	1	0	1	1	0	0	1	
1	1	0	0	0	0	1	0	1	1	1	0	1	0	
0	0	1	0	1	1	0	1	0	0	1	1	1	1	

Simulation

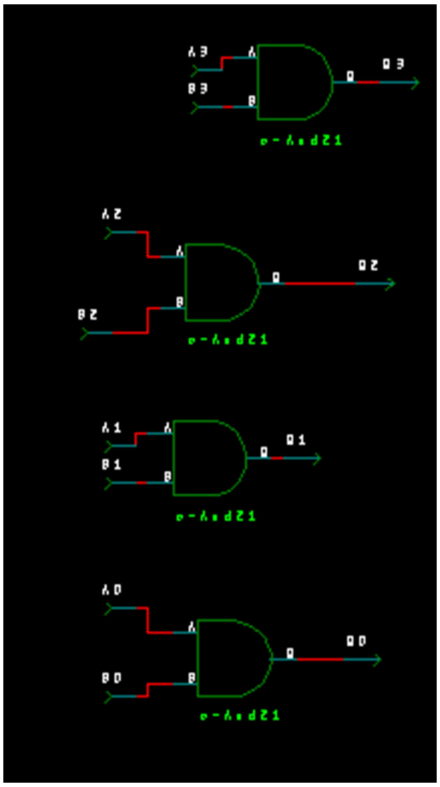


4 bit and gate

Symbol



Schematic



Command File

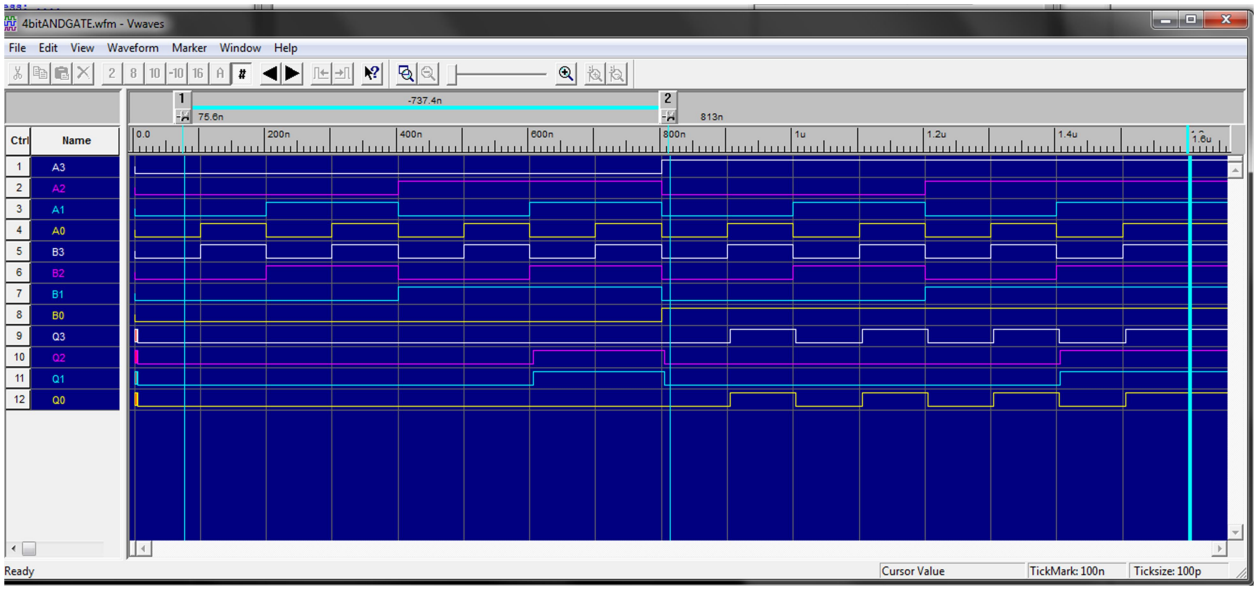
```
restart
wave 4bitANDGATE.wfm A3 A2 A1 A0 B3 B2 B1 B0 Q3 Q2 Q1 Q0
pattern A3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
pattern A2 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
pattern A1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
pattern A0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
pattern B3 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
pattern B2 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
pattern B1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
pattern B0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
run
```

Expected Results

4 bit and gate

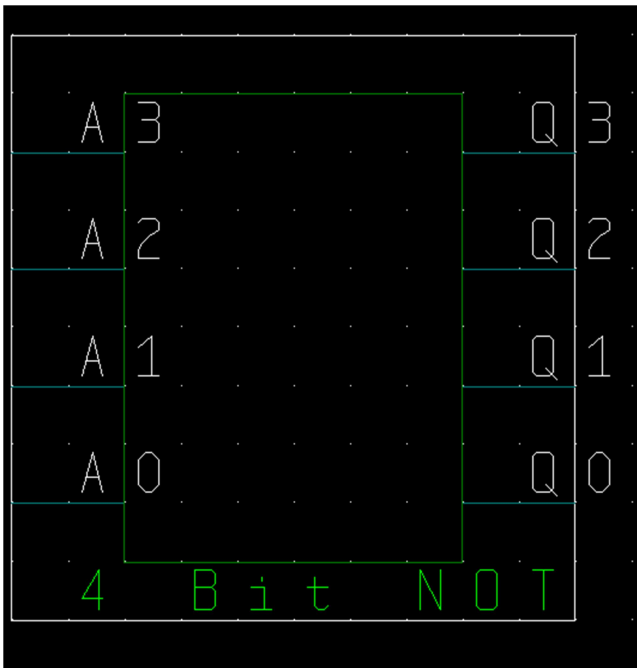
A3	A2	A1	A0	B3	B2	B1	B0	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1	0	0	0	0
1	0	0	1	1	0	0	1	1	0	0	1
1	0	1	0	0	1	0	1	0	0	0	0
1	0	1	1	1	1	0	1	1	0	0	1
1	1	0	0	0	0	1	1	0	0	0	0
1	1	0	1	1	0	1	1	1	0	0	1
1	1	1	0	0	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1

Simulation

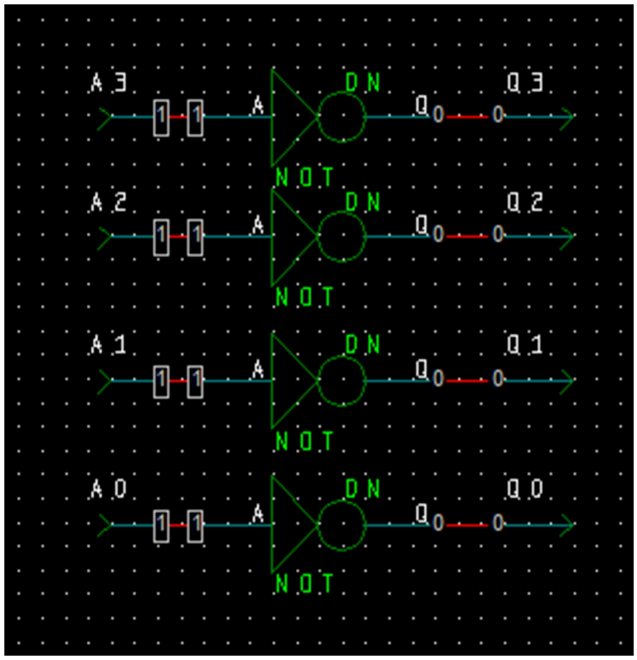

4bitANDGATE.wfm - Vwaves
File Edit View Waveform Marker Window Help
-737.4n
75.6n
813n
Ctrl
Name
A3
A2
A1
A0
B3
B2
B1
B0
Q3
Q2
Q1
Q0
0.0
200n
400n
600n
800n
1u
1.2u
1.4u
Ready
Cursor Value
TickMark: 100n
Ticksize: 100p

4 bit not gate

Symbol



Schematic



Command File

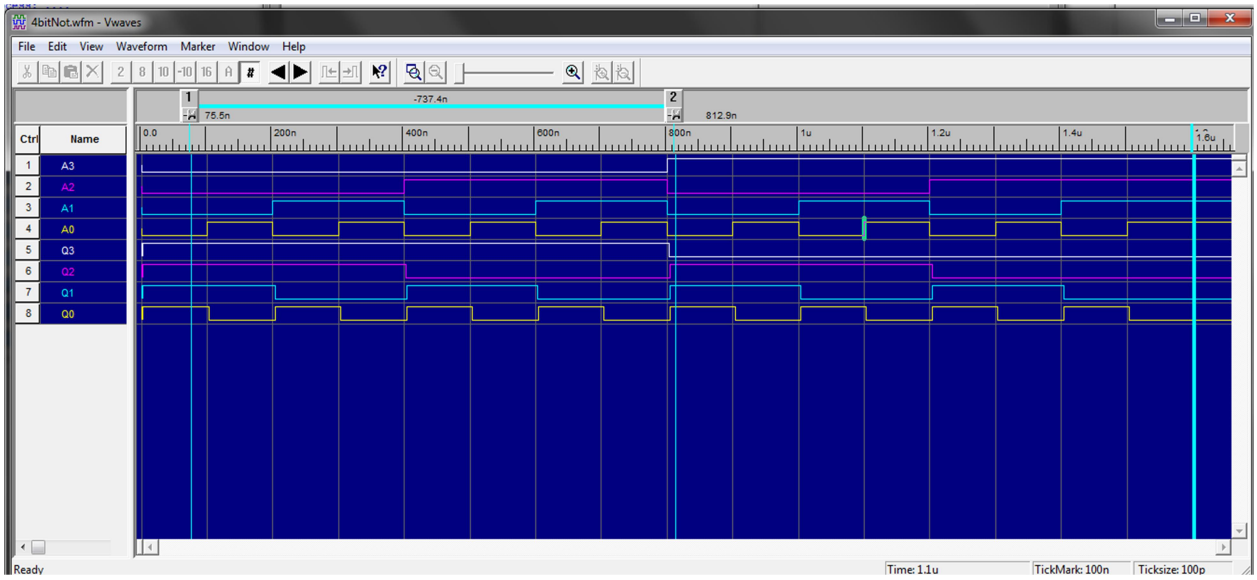
```
restart
wave 4bitNot.wfm A3 A2 A1 A0 Q3 Q2 Q1 Q0
pattern A3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
pattern A2 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
pattern A1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
pattern A0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
run
```

Expected Results

4 bit not gate

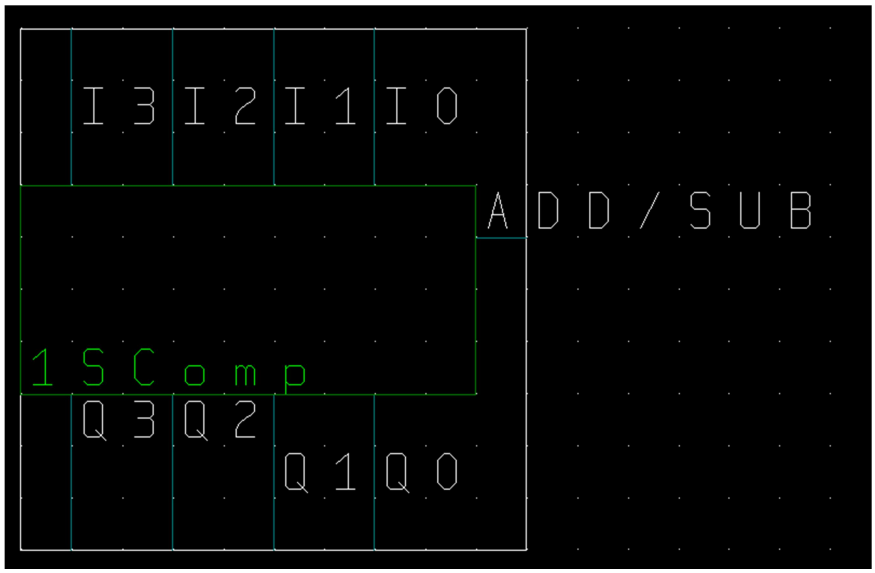
A3	A2	A1	A0	Q3	Q2	Q1	Q0
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

Simulation

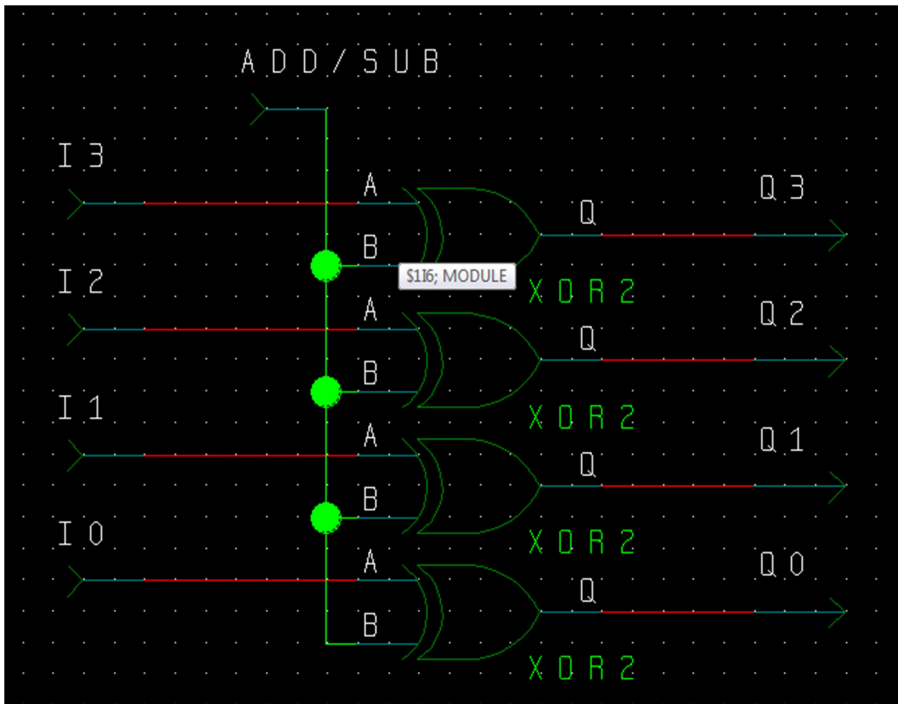

4bitNot.wfm - Vwaves
File Edit View Waveform Marker Window Help
-737.4n
75.5n
812.9n
Ctrl
Name
A3
A2
A1
A0
Q3
Q2
Q1
Q0
0.0
200n
400n
600n
800n
1u
1.2u
1.4u
Ready
Time: 1.1u
TickMark: 100n
Ticksize: 100p

1s compliment

Symbol



Schematic



Command File

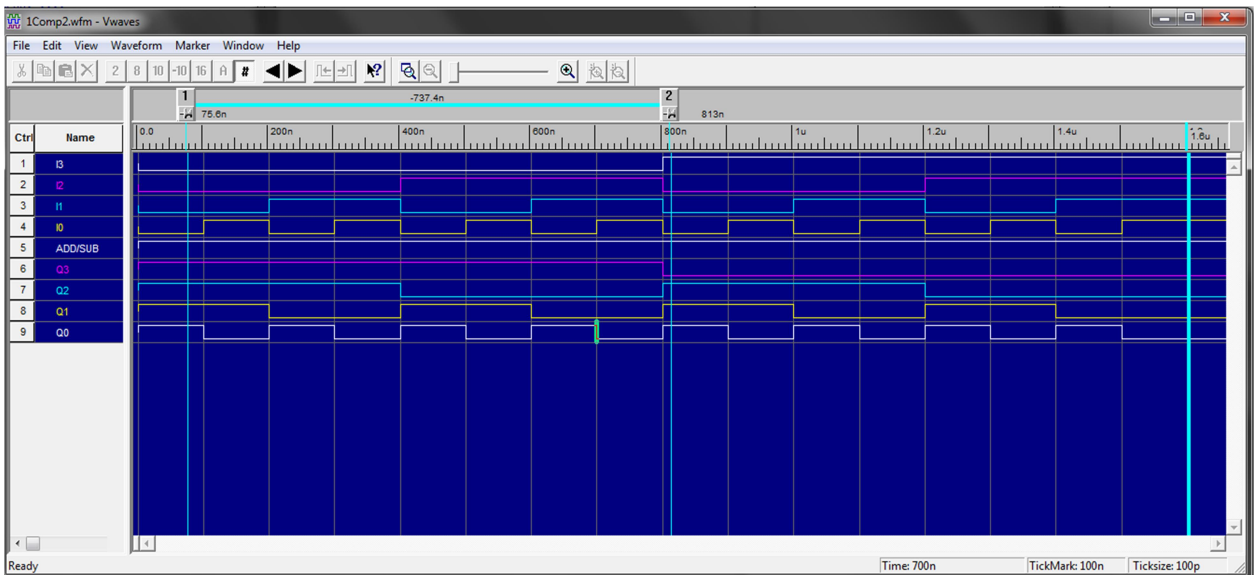
```
restart
wave 1Comp2.wfm I3 I2 I1 I0 ADD/SUB Q3 Q2 Q1 Q0
pattern I3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
pattern I2 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
pattern I1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
pattern I0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
pattern 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
run
```

Expected Results

1's Complement

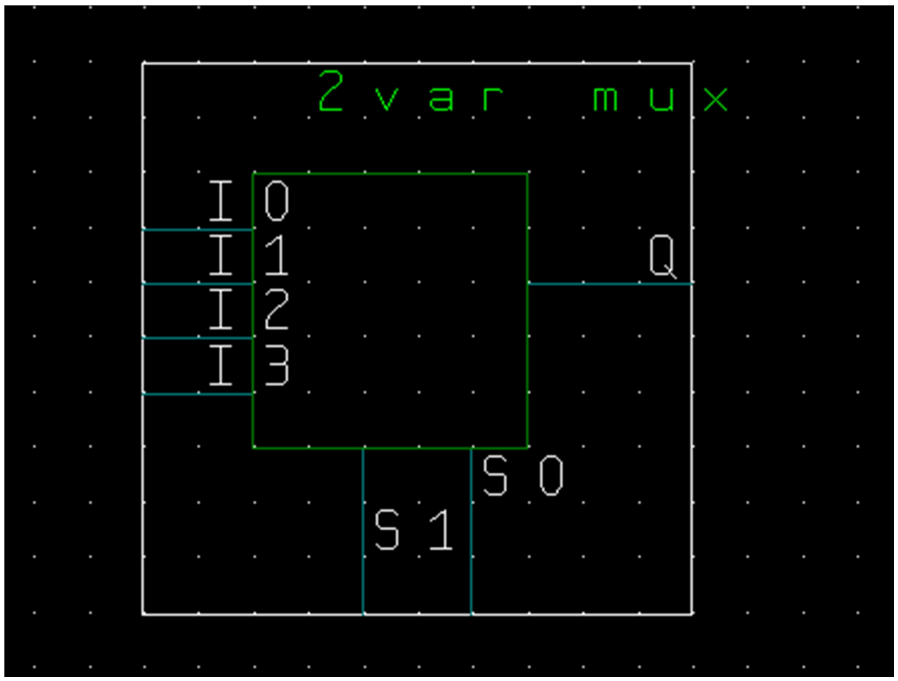
I3	I2	I1	I0	Q3	Q2	Q1	Q0	ADD/SUB
0	0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	0	1
0	0	1	0	1	1	0	1	1
0	0	1	1	1	1	0	0	1
0	1	0	0	1	0	1	1	1
0	1	0	1	1	0	1	0	1
0	1	1	0	1	0	0	1	1
0	1	1	1	1	0	0	0	1
1	0	0	0	0	1	1	1	1
1	0	0	1	0	1	1	0	1
1	0	1	0	0	1	0	1	1
1	0	1	1	0	1	0	0	1
1	1	0	0	0	0	1	1	1
1	1	0	1	0	0	1	0	1
1	1	1	0	0	0	0	1	1
1	1	1	1	0	0	0	0	1

Simulation

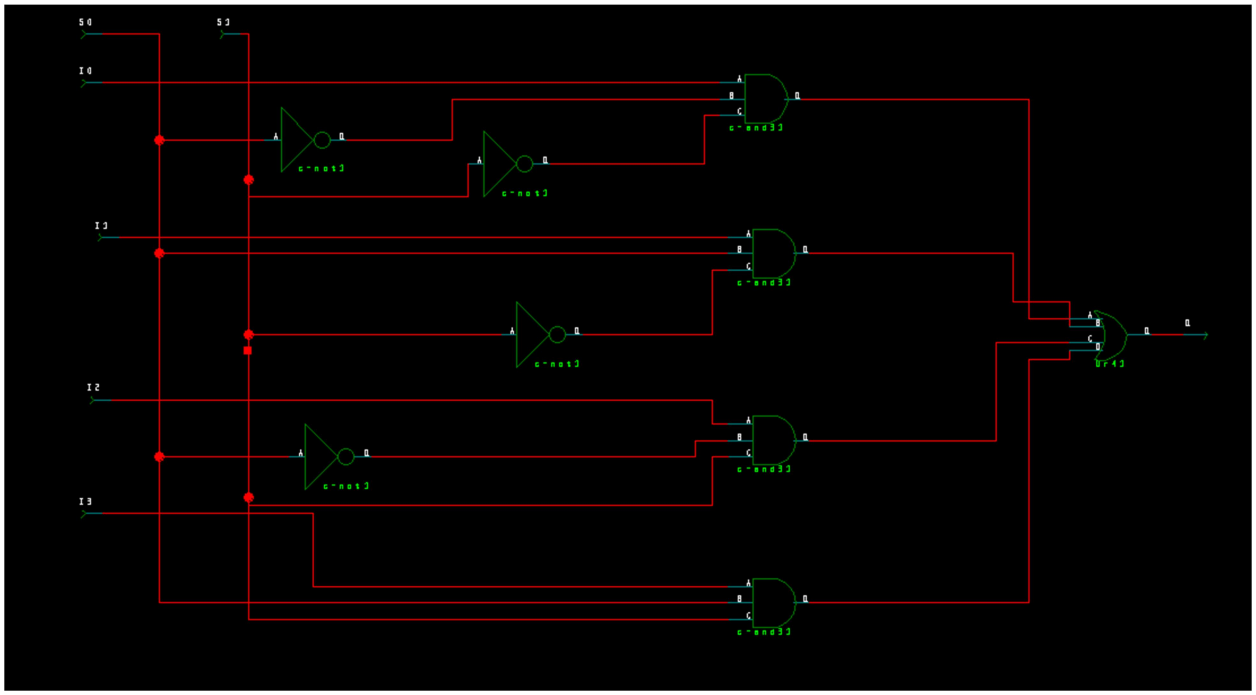


2-Var Mux

Symbol



Schematic



Command File

restart

wave 2varMux.wfm S0 S1 I0 I1 I2 I3 Q

pattern S0 0 1 0 1

pattern S1 0 0 1 1

pattern I0 1 0 0 0

pattern I1 0 1 0 0

pattern I2 0 0 1 0

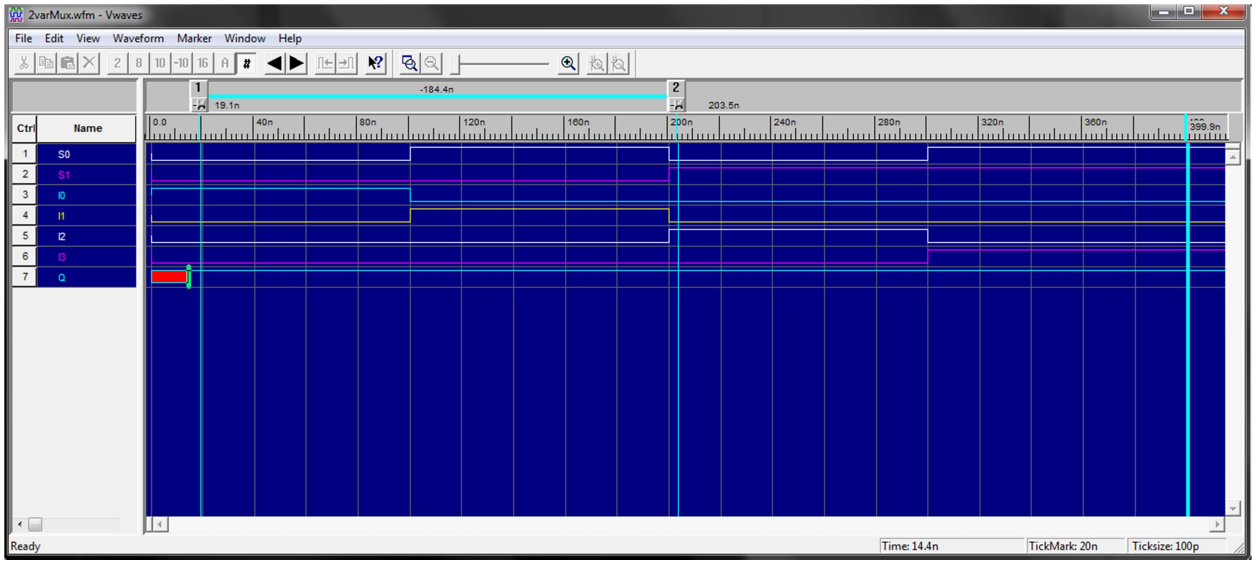
pattern I3 0 0 0 1

run

Expected Results

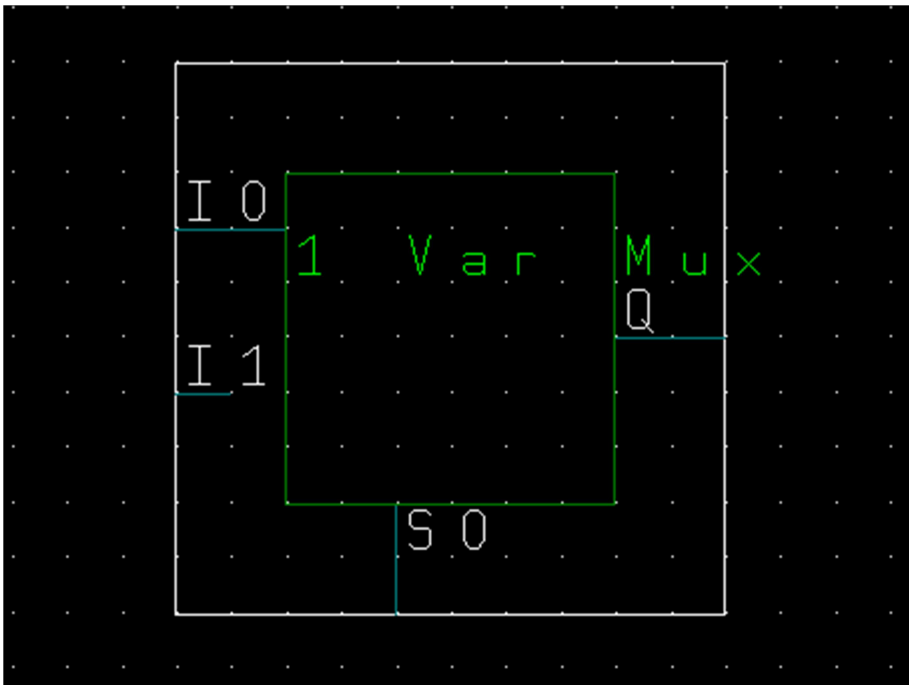
S0	S1	I0	I1	I2	I3	Q	2 Var Mux
0	0	1	0	0	0	1	
1	0	0	1	0	0	1	
0	1	0	0	1	0	1	
1	1	0	0	0	1	1	

Simulation

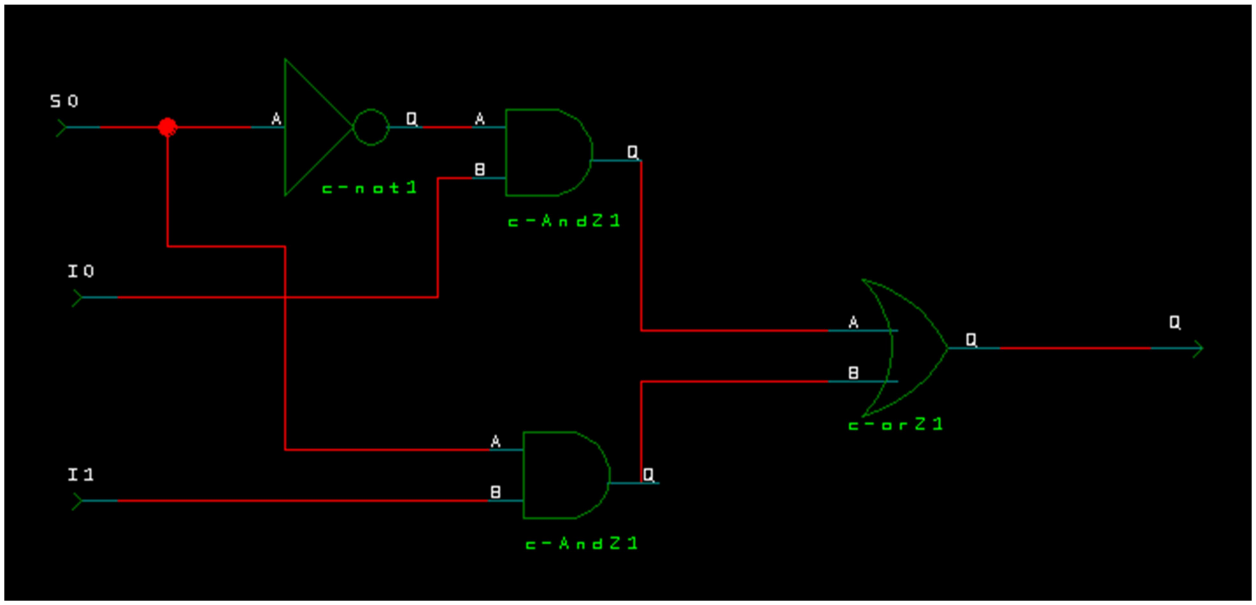


1-Var Mux

Symbol



Schematic



Command File

restart

wave 2varMux.wfm S0 S1 I0 I1 I2 I3 Q

pattern S0 0 1 0 1

pattern S1 0 0 1 1

pattern I0 1 0 0 0

pattern I1 0 1 0 0

pattern I2 0 0 1 0

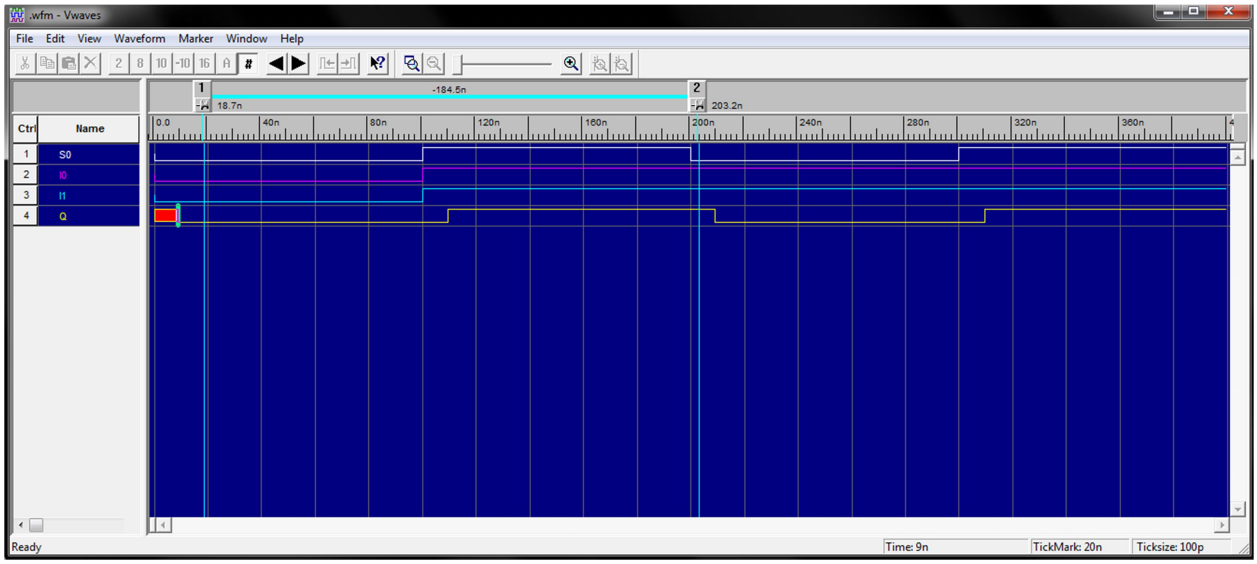
pattern I3 0 0 0 1

run

Expected Results

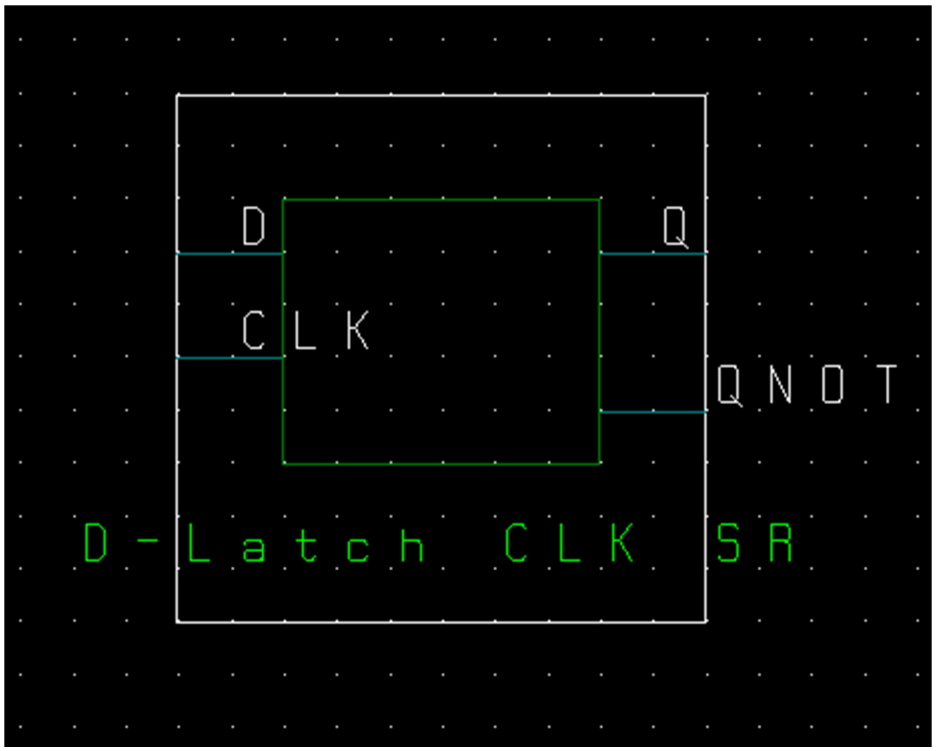
S0	I0	I1	Q	1 Var mux
0	1	0	1	
1	0	1	1	
0	0	1	0	
1	1	1	1	

Simulation

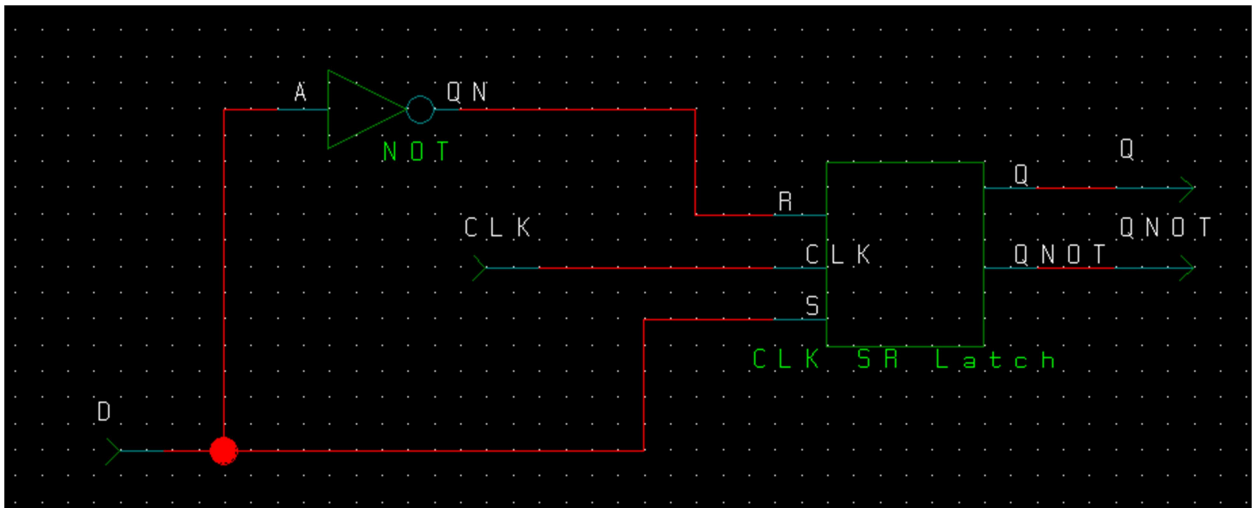


D-Latch

Symbol



Schematic



Command File

restart

wave d-latch-clk-sr.wfm D CLK Q QNOT

pattern D 0 0 1 1 0 1 1 1 0

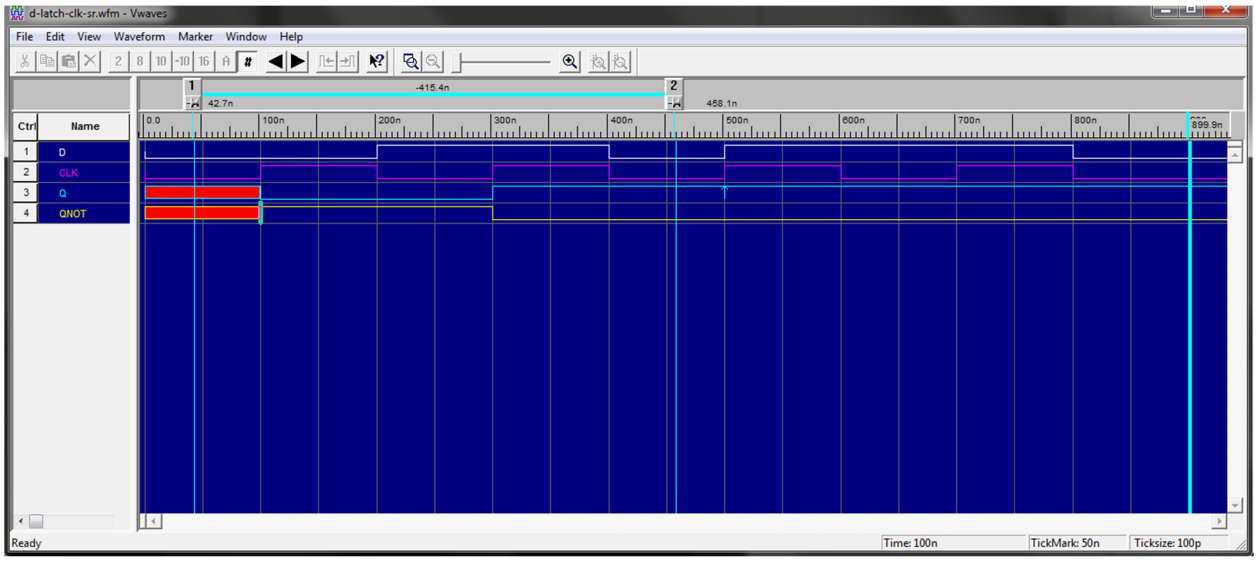
pattern CLK 0 1 0 1 0 1 0 1 0

run

Expected Results

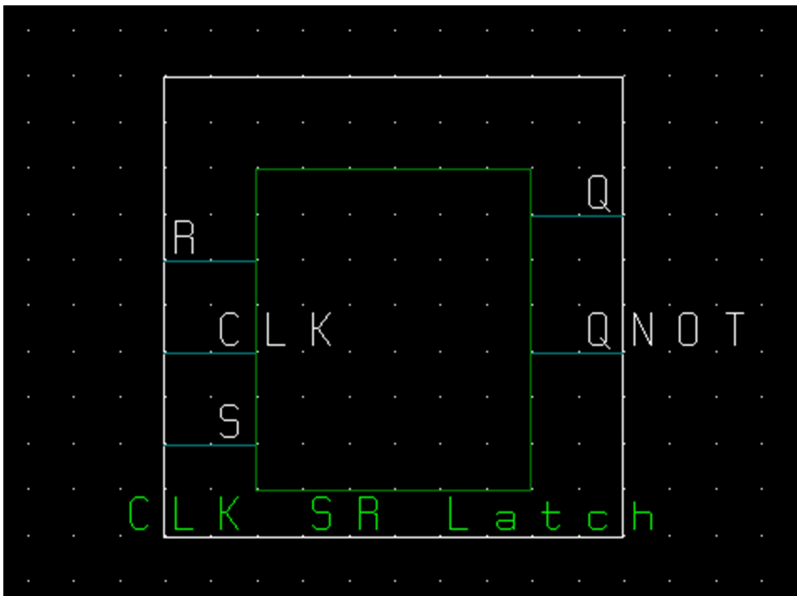
S	R	Q	QNOT	CLK SR LATCH
0	0	0	1	
0	1	0	1	
1	0	0	1	
1	1	0	0	

Simulation

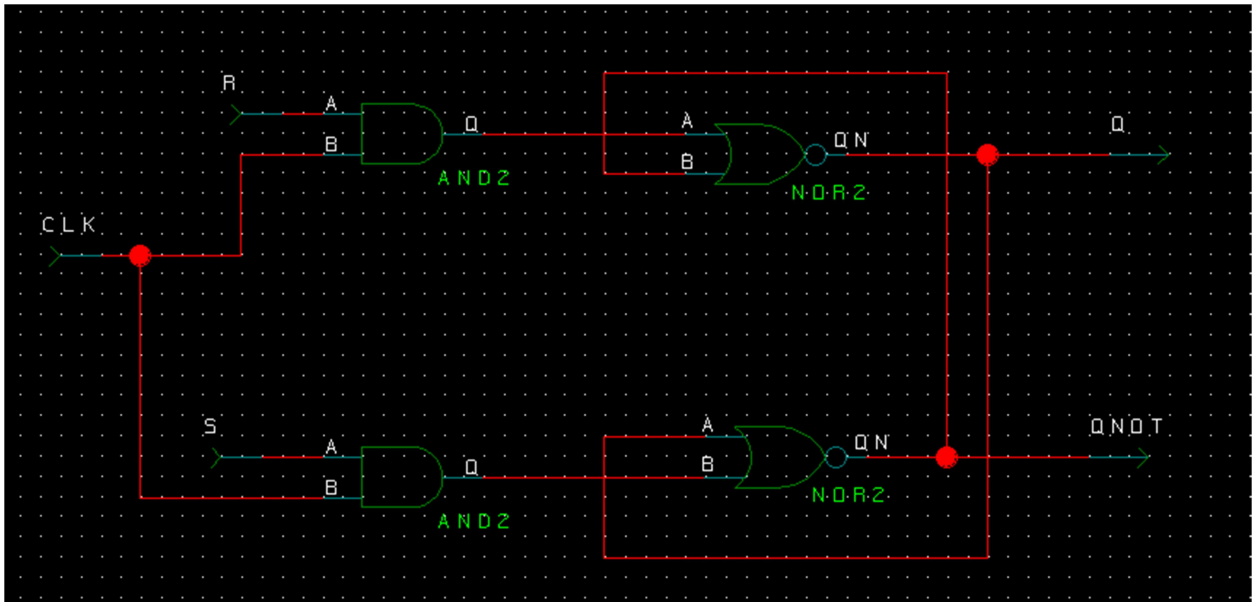


CLK SR-Latch

Symbol



Schematic



Command File

restart

wave S-RLacth.wfm S R Q NQ

pattern S 0 0 1 1

pattern R 0 1 0 1

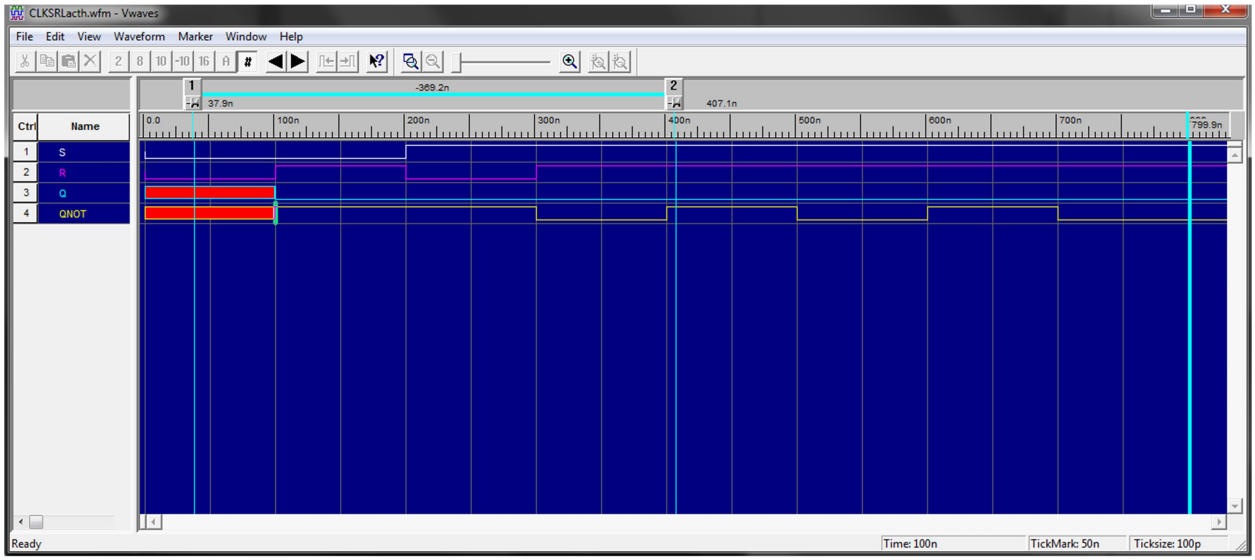
pattern D 0 0 1 1 0 1 1 1 0 (D Latch)

run

Expected Results

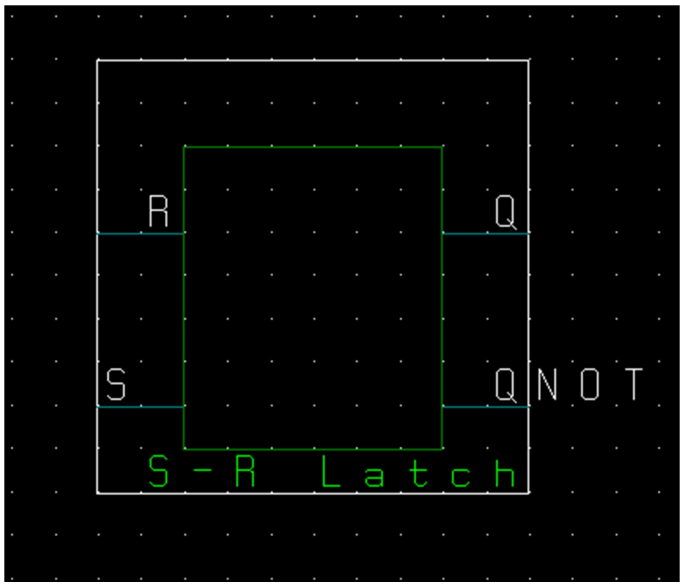
S	R	Q	QNOT	CLK SR LATCH
0	0	0	1	
0	1	0	1	
1	0	0	1	
1	1	0	0	

Simulation

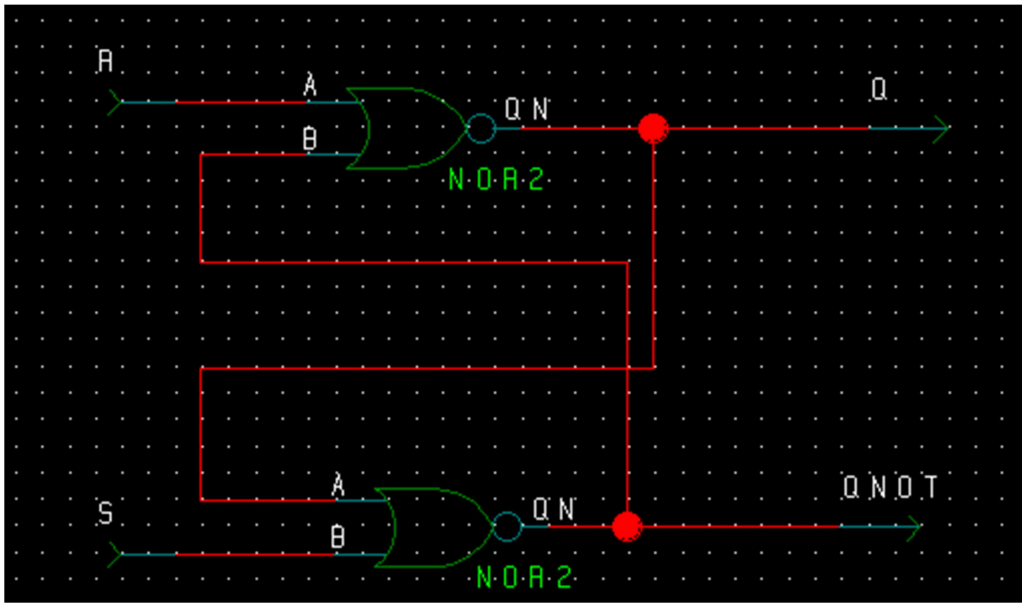


SR Latch

Symbol



Schematic



Command File

restart

wave SRLatch.wfm S R Q

pattern S 0 0 1 1

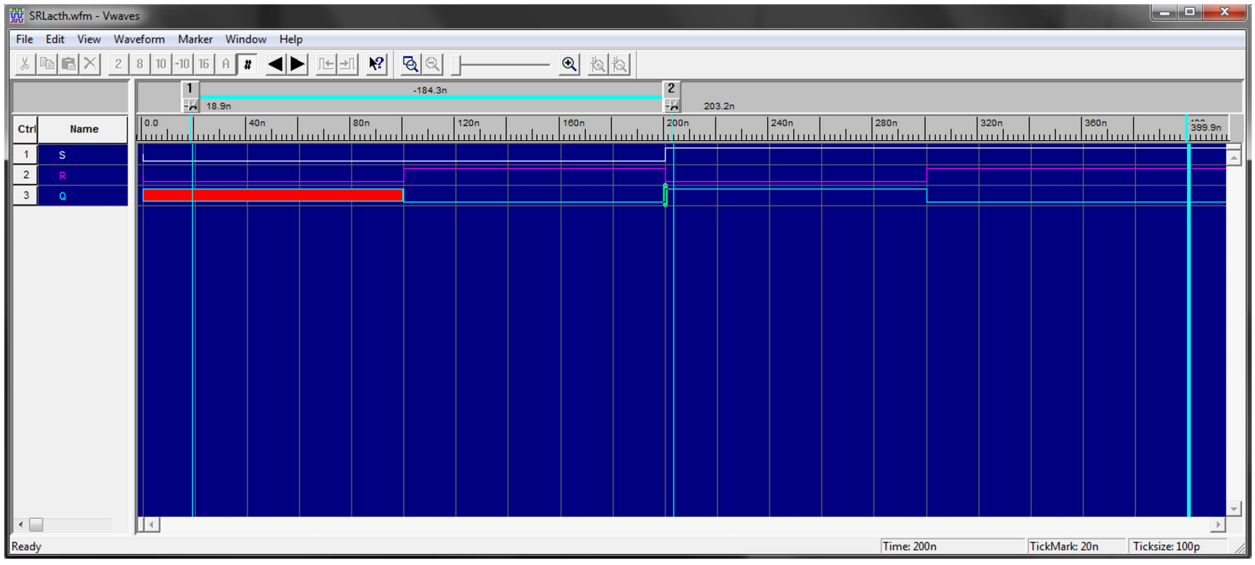
pattern R 0 1 0 1

run

Expected Results

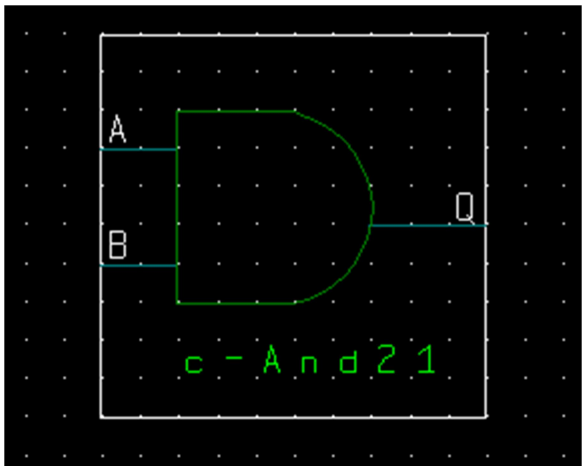
S	R	Q	SR LACTH
0	0	0	
0	1	1	
1	0	0	
1	1	1	

Simulation

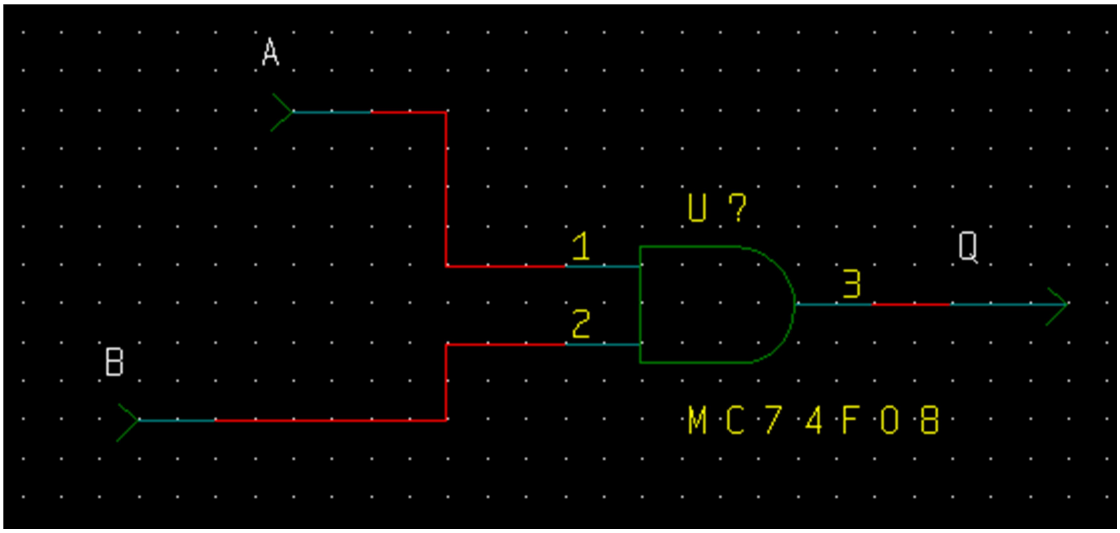


C and 2-1

Symbol



Schematic



Command File

restart

wave c-and21.wfm A B Q

pattern A 0 0 1 1

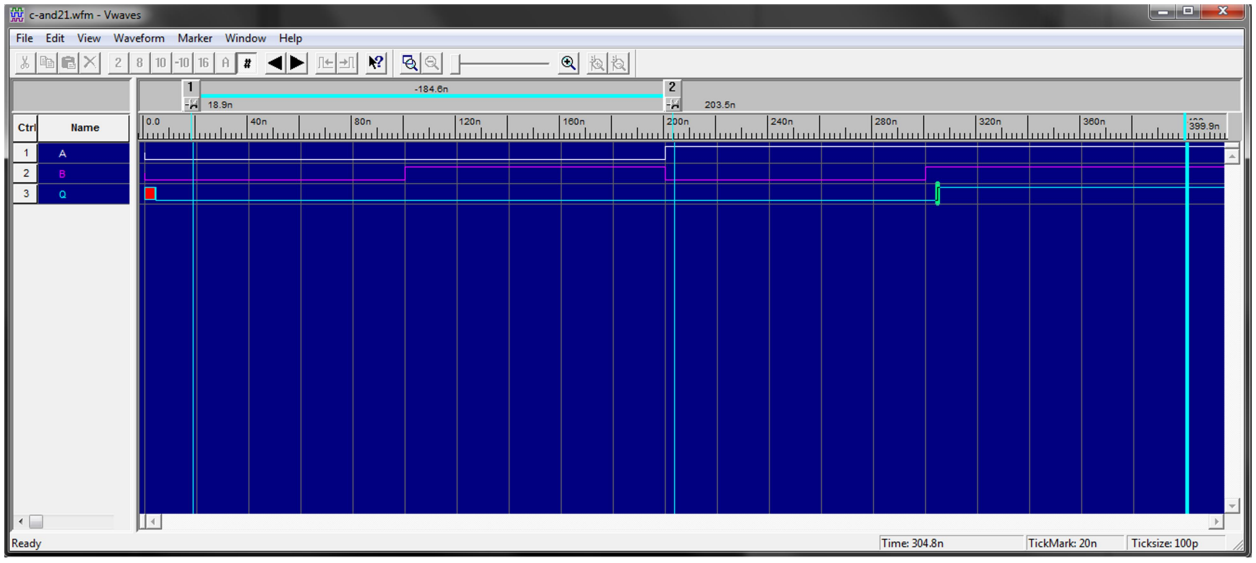
pattern B 0 1 0 1

run

Expected Results

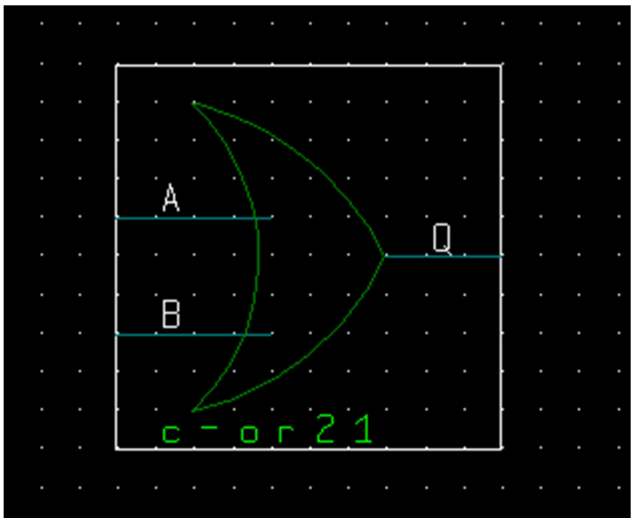
A	B	Q	
0	0	0	AND
0	1	0	
1	0	0	
1	1	1	

Simulation

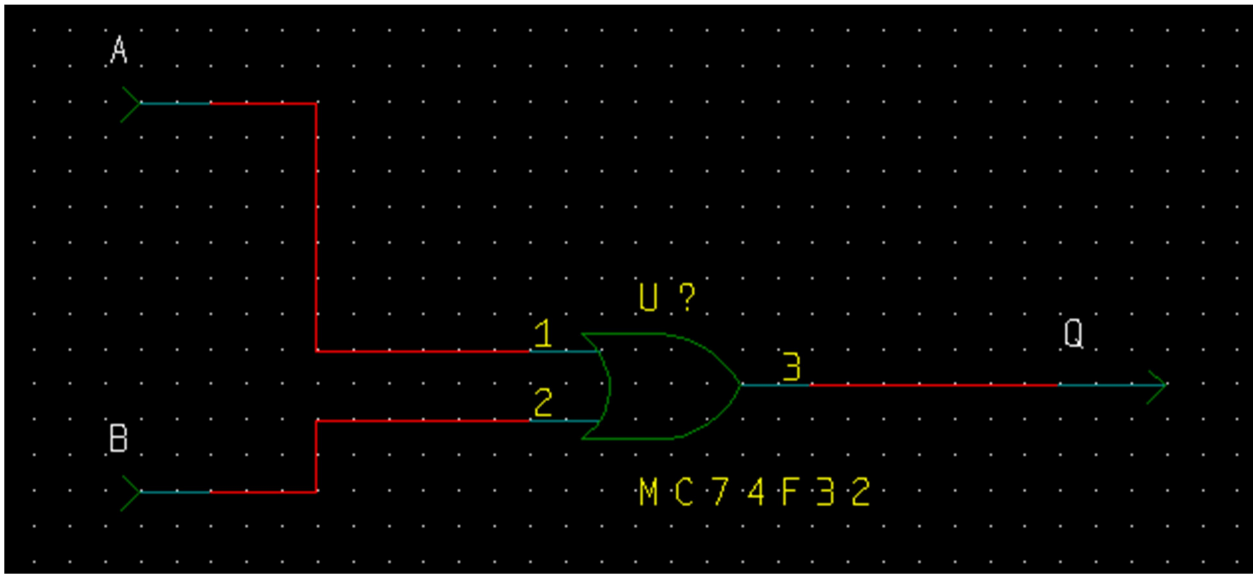


C or 2-1

Symbol



Schematic



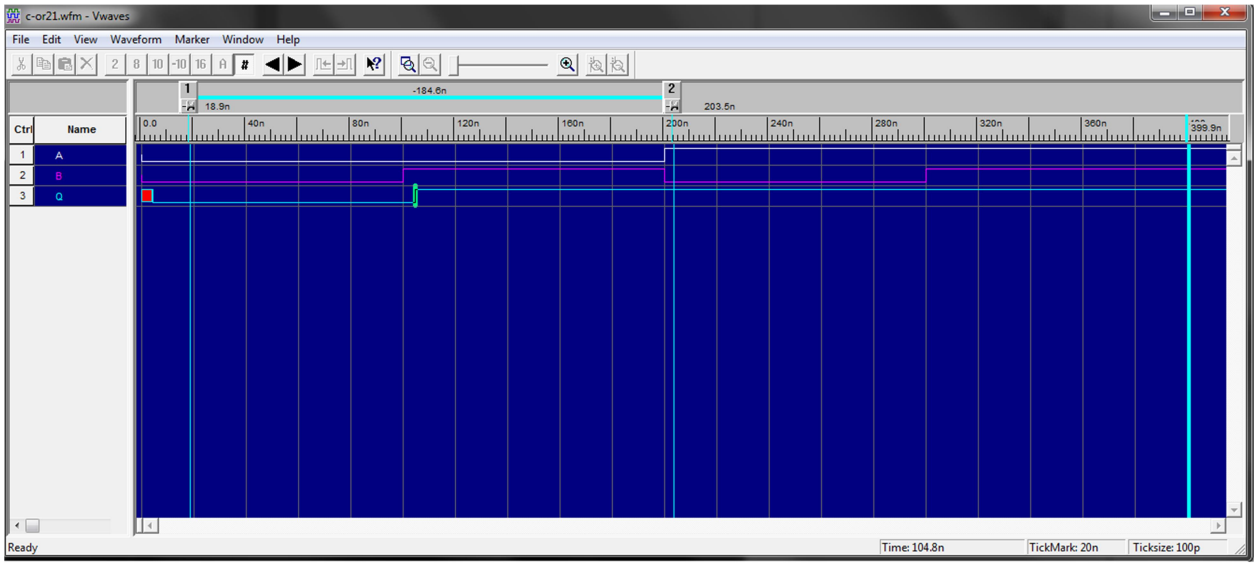
Command File

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

Expected Results

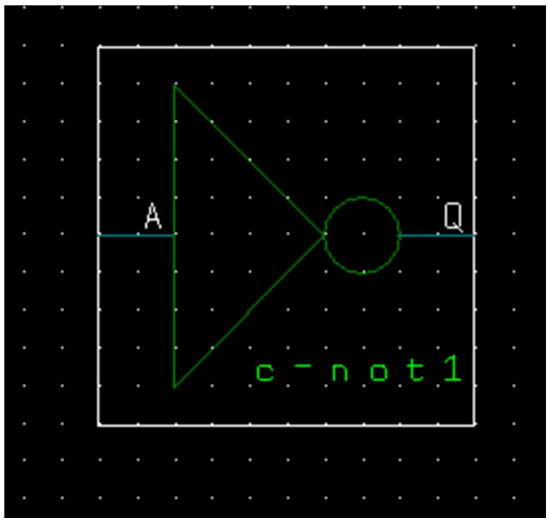
A	B	Q	OR
0	0	0	
0	1	1	
1	0	1	
1	1	1	

Simulation

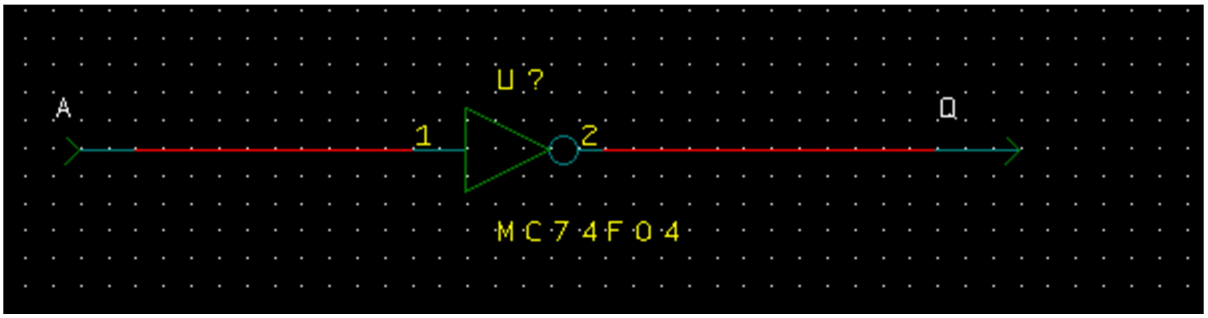


C not 1

Symbol



Schematic



Command File

restart

wave c-not1.wfm A Q

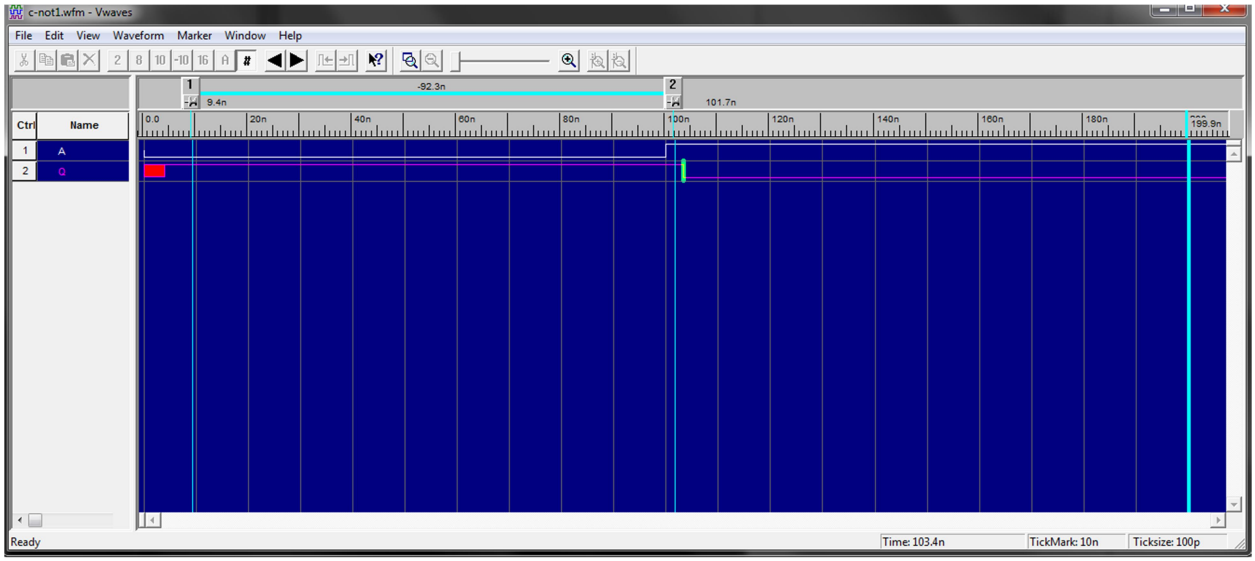
pattern A 0 1

run

Expected Results

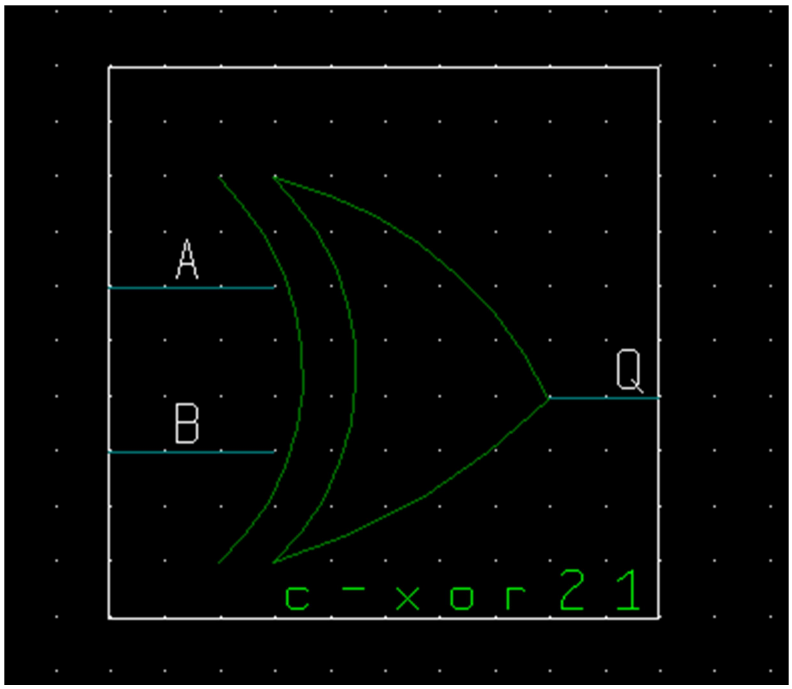
A	Q	NOT
0	1	
1	0	

Simulation

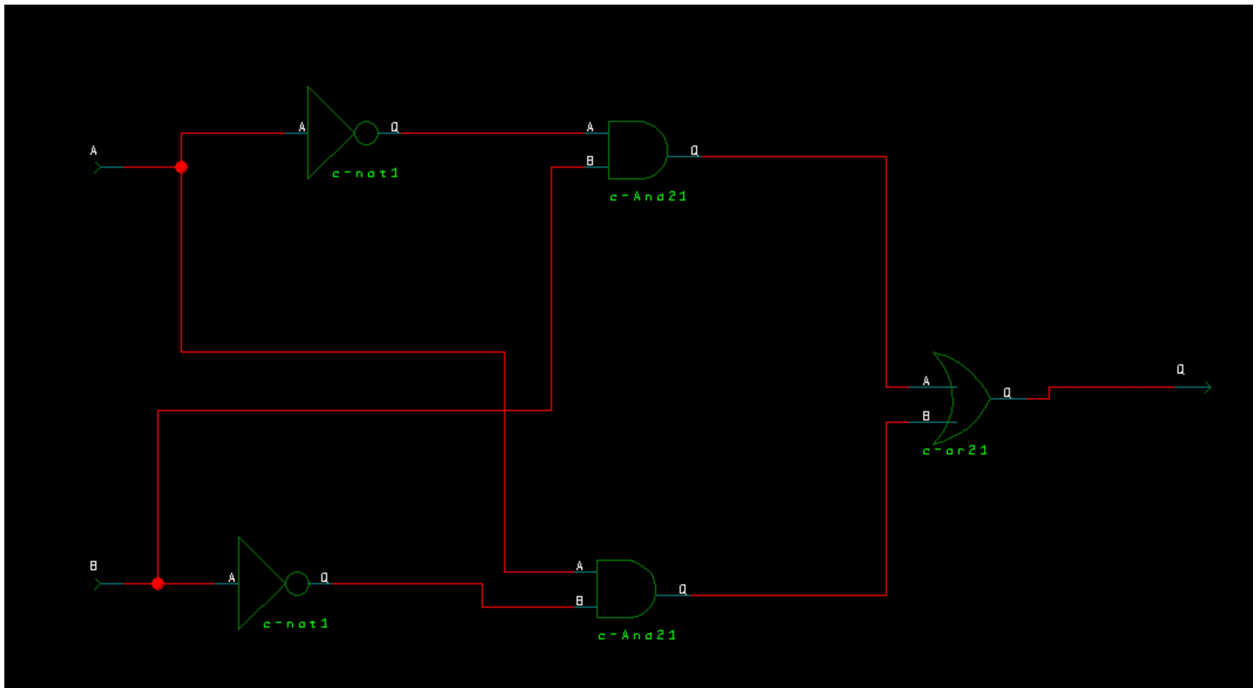


C xor 2-1

Symbol



Schematic



Command File

restart

wave c-xor21.wfm A B Q

pattern A 0 0 1 1

pattern B 0 1 0 1

run

Expected Results

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

Simulation