CS220 Computer Architecture
Digital Logic Design
Practical 4

The TkGate Logic Simulator under Linux is used to implement these practicals.
Boot Linux, log in and launch TkGate. The following circuits were covered in lectures 6,7,8 and the adder circuit was also constructed in Lab 3. Slides are available on moodle.

Part A

Construct a 2-bit adder circuit by composing two 1-bit adders with additional logic so that it can act both as a 2-bit adder and as a 2-bit subtractor. Use the 5 gate adder design seen already for each 1-bit adder. You can use last week's lab file and save as a different name to get started. The two 2-bit numbers will be supplied by four switches.

Two extra signals, ADD and SUBTRACT can be provided by switches. When ADD is on, the circuit should act as an adder. When SUBTRACT is on, the circuit should act as a two's complement subtractor. Assume only the ADD or SUBTRACT signal will be active, but not both together. The logic required was given on slide 40 of lecture 5 and 6.

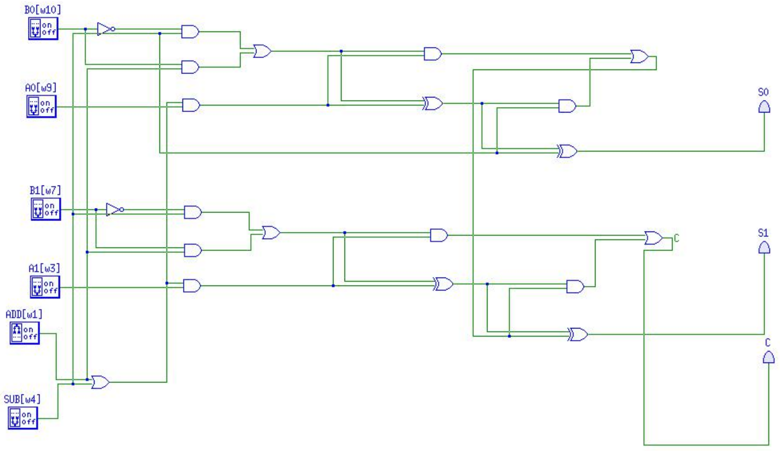
Note that when using 2's complement arithmetic for subtraction on 2-bit numbers the range of valid numbers that can be used are 11 for -1, 00 for 0 and 01 for 1. Applying numbers outside this range may lead to unexpected results.

Try A=01 and B=01 for Subtract, answer = 00. Try A=11 and B=11 for Subtract, answer = 00.
i.e. 1-1=0 and -1–(-1) = 0.
Please ensure to have your work done at least 15 mins prior to the end to allow time for correcting.

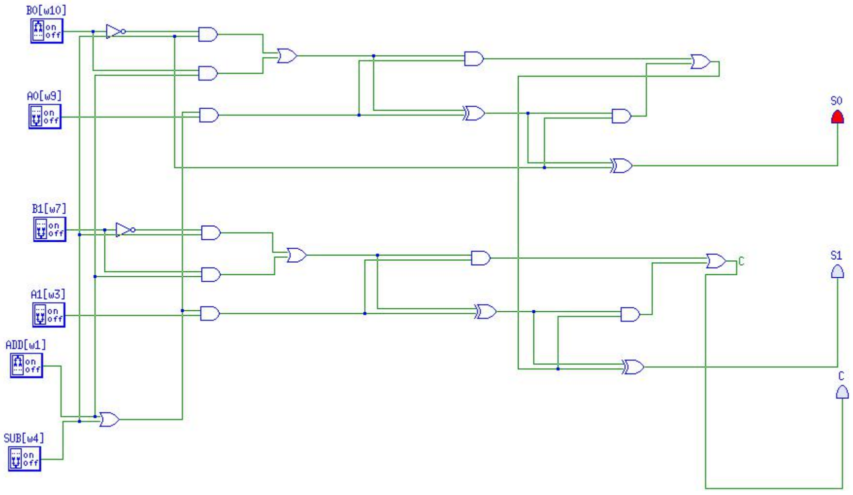
ADD： $A+B$

A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	2
0	0	1	1	0	1	1	3
1	0	0	0	0	1	0	2
1	0	0	1	0	1	1	3
1	0	1	0	1	0	0	4
1	0	1	1	1	0	1	5
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	2
0	1	1	0	0	1	1	3
0	1	1	1	1	0	0	4
1	1	0	0	0	1	1	3
1	1	0	1	1	0	0	4
1	1	1	0	1	0	1	5
1	1	1	1	1	1	0	6

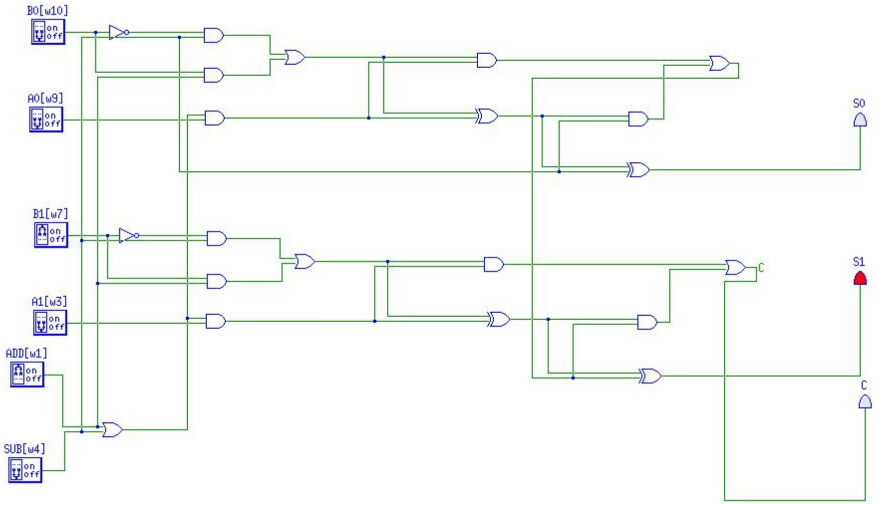
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	0	0	0	0	0



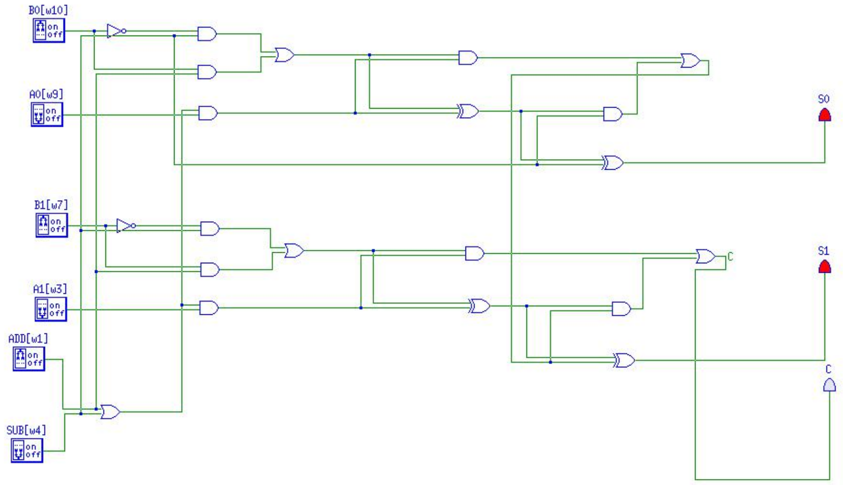
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	1	0	0	1	1



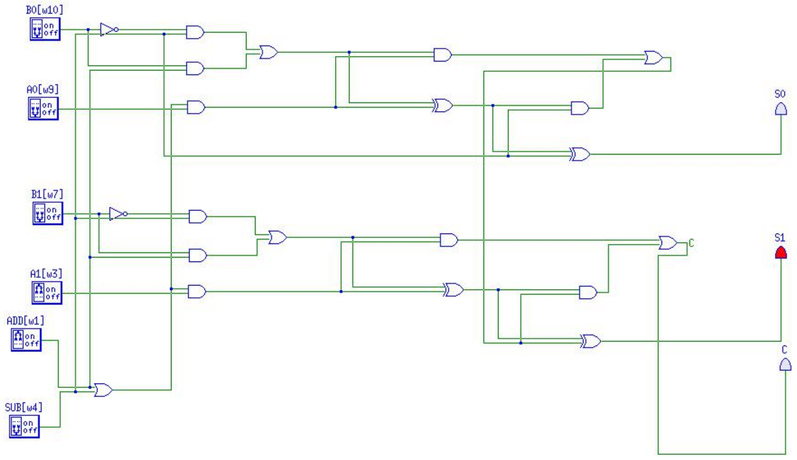
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	1	0	0	1	0	2



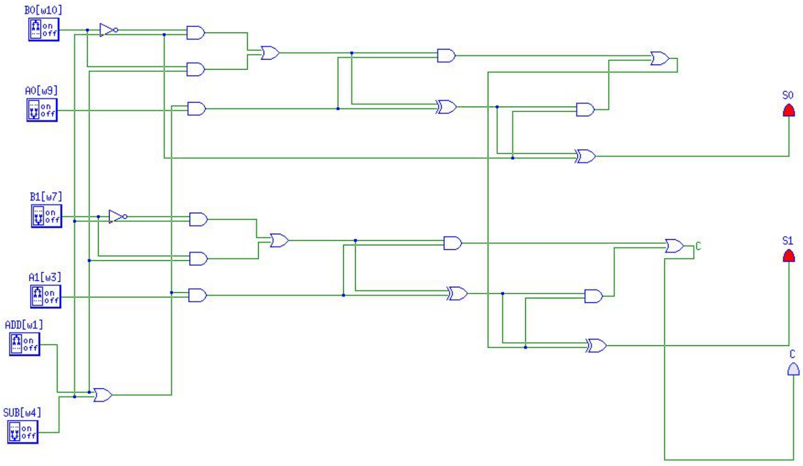
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	1	1	0	1	1	3



A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
1	0	0	0	0	1	0	2



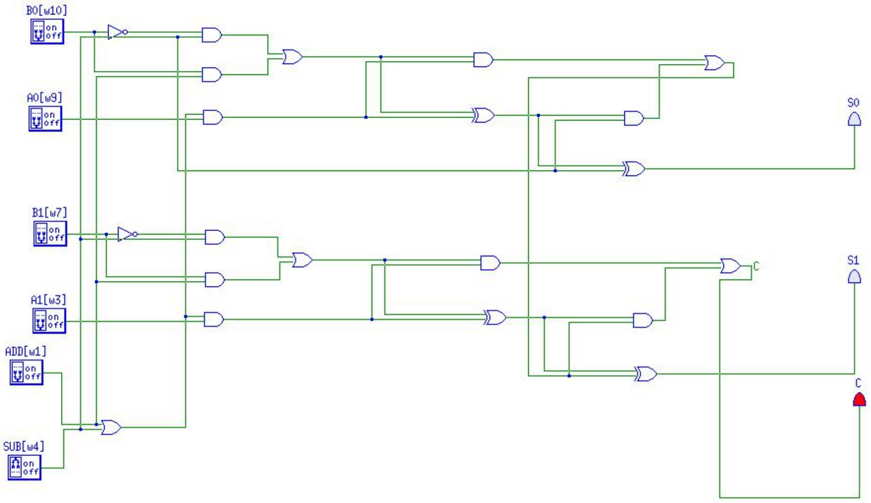
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
1	0	0	1	0	1	1	3



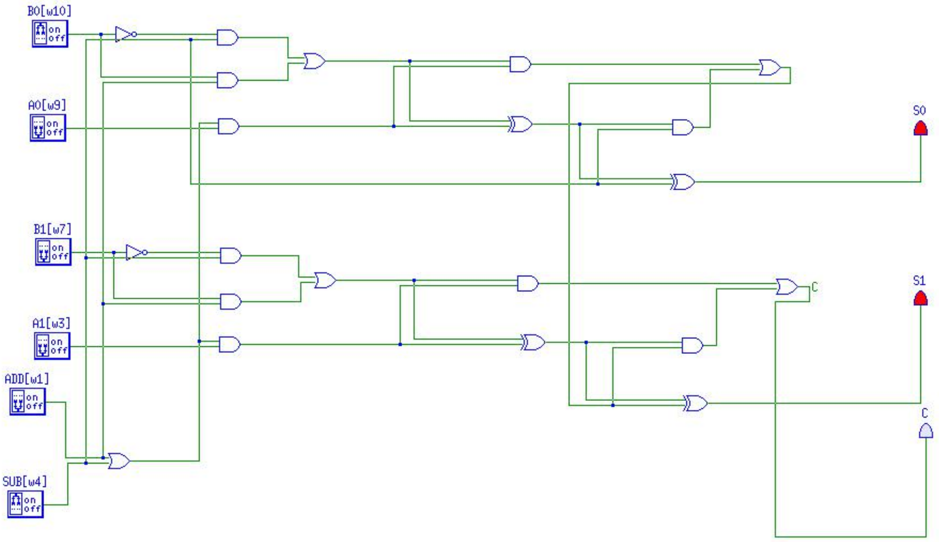
SUBTRACT：$A-B$

A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	-1
0	0	1	1	0	0	1	1
0	1	0	0	1	0	1	1
0	1	0	1	1	0	0	0
0	1	1	1	0	1	0	2
1	1	0	0	1	1	1	-1
1	1	0	1	1	1	0	-2
1	1	1	1	1	0	0	0

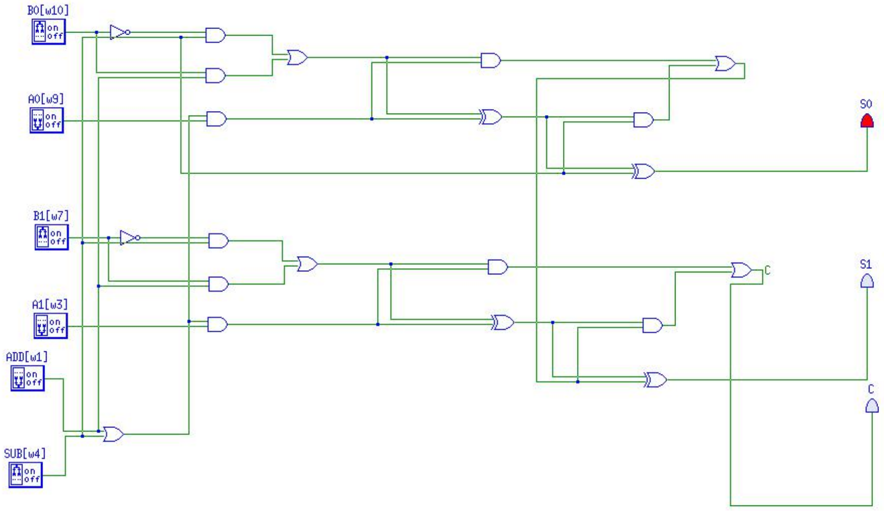
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	0	1	0	0	0



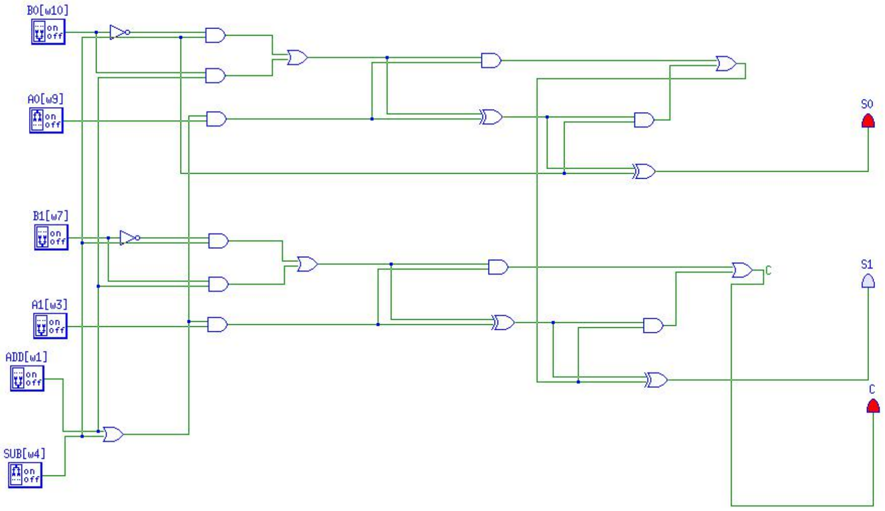
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	0	1	0	1	1	-1



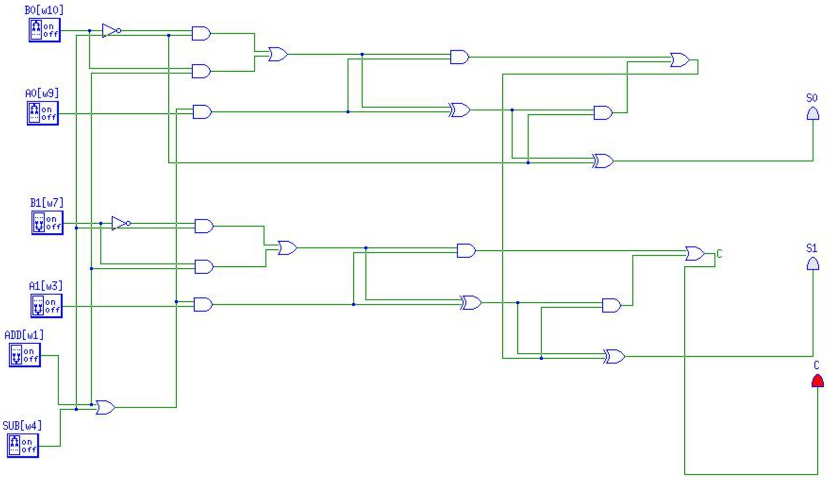
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	0	1	1	0	0	1	1



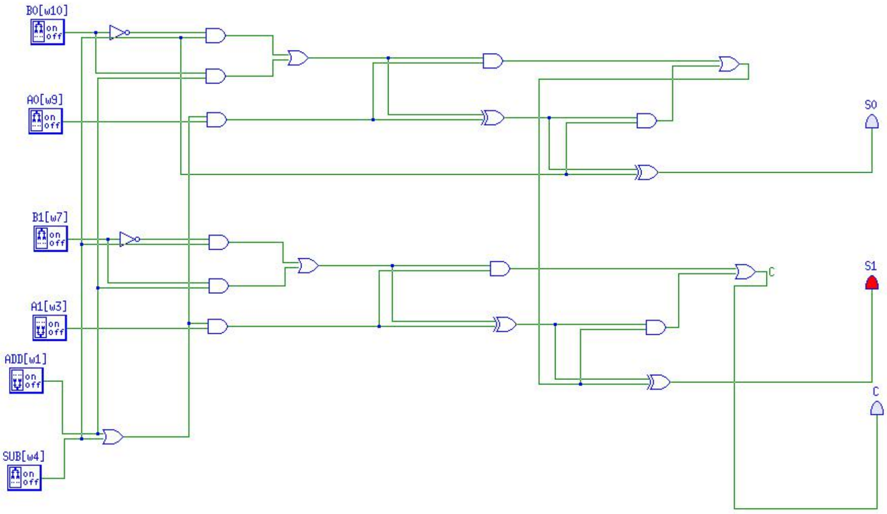
A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	1	0	0	1	0	1	1



A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	1	0	1	1	0	0	0



A_1	A_0	B_1	B_0	C_0	S_1	S_0	V
0	1	1	1	0	1	0	2



Part B

Implement the four variable switching function f(A,B,C,D) = m (0,4,5,8,9,10,12) using an 8-1 Multiplexer and verify that the circuit obeys the truth table for this function.

Notes:

The multiplexer is available as a component in TKGate and does not have to be implemented at the gate level.

The four variables should be represented as switches. Three switches (A,B,C) will be connected to the control inputs of the multiplexer.

The eight inputs to the multiplexer will be connected in turn either to logic 0 (GND), logic 1 (Vdd), Switch D or the complement of switch D as determined from the function output requirements. You should depict these on a truth table.

Note the control inputs to the multiplexer are depicted as a single red wire in the TKGate Simulator. You will need to create a **wire merge device** (from the I/O device menu). Then double click on the wire merge device and add a third (green wire) input port. The red wire can be connected to the multiplexer control inputs and the switches representing A,B and C can be connected to the wire merge device ports 2,1 and 0 respectively.

The true table for f(A,B,C,D) = m (0,4,5,8,9,10,12)

A	***B***	***C***	***D***	***f(A,B,C,D)***
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

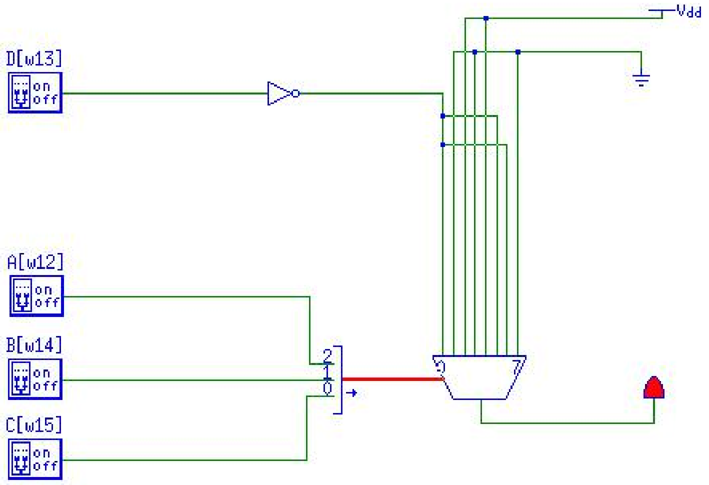
K-MAP:

AB CD	**00**	**01**	**11**	**10**
00	**1**	**1**	**1**	**1**
01	**0**	**1**	**0**	**1**
11	**0**	**0**	**0**	**0**
10	**0**	**0**	**0**	**1**

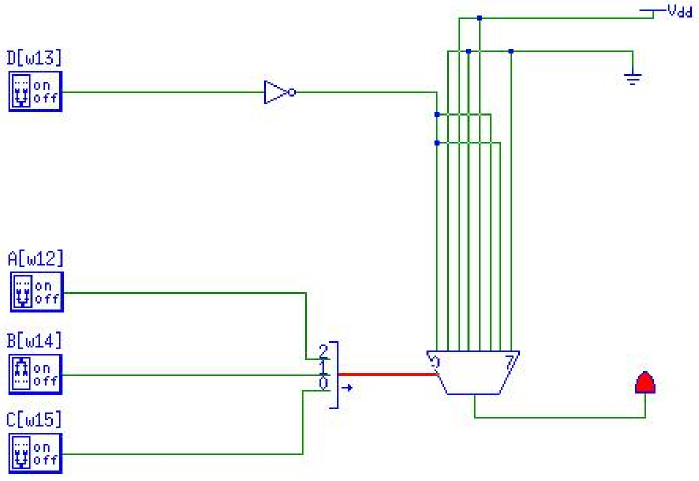
The design of the 8-1 Multiplexer:

	A	***B***	***C***	***INPUT***
D_0	0	0	0	$\overline{D}$
D_1	0	0	1	GND
D_2	0	1	0	VDD
D_3	0	1	1	GND
D_4	1	0	0	VDD
D_5	1	0	1	$\overline{D}$
D_6	1	1	0	$\overline{D}$
D_7	1	1	1	GND

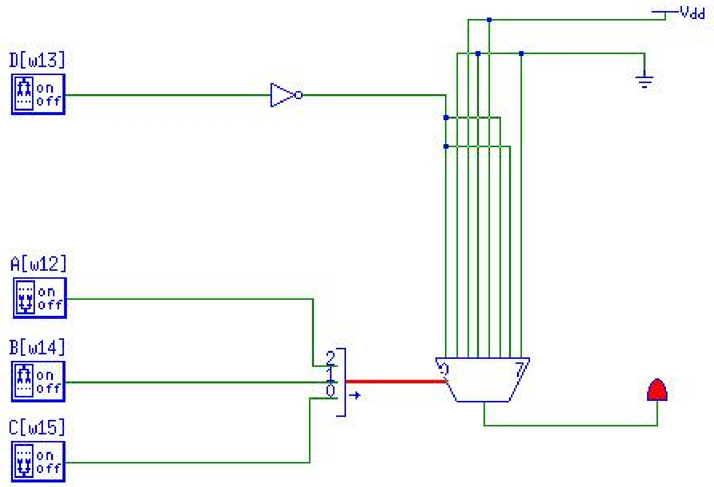
A	***B***	***C***	***D***	***f(A,B,C,D)***
0	0	0	0	1



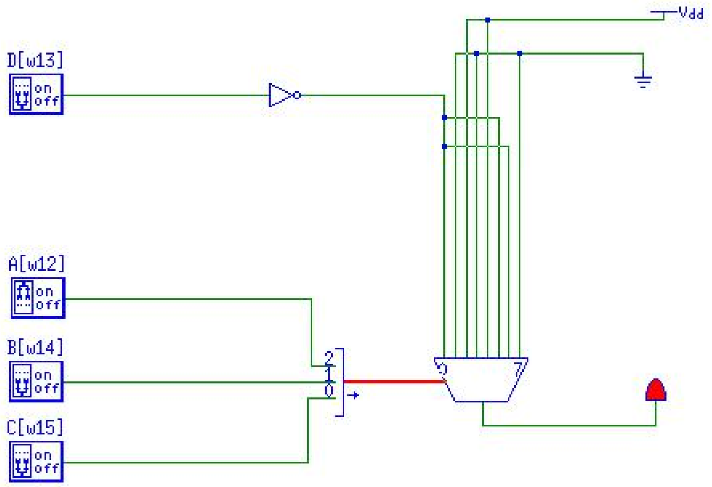
A	***B***	***C***	***D***	***f(A,B,C,D)***
0	1	0	0	1



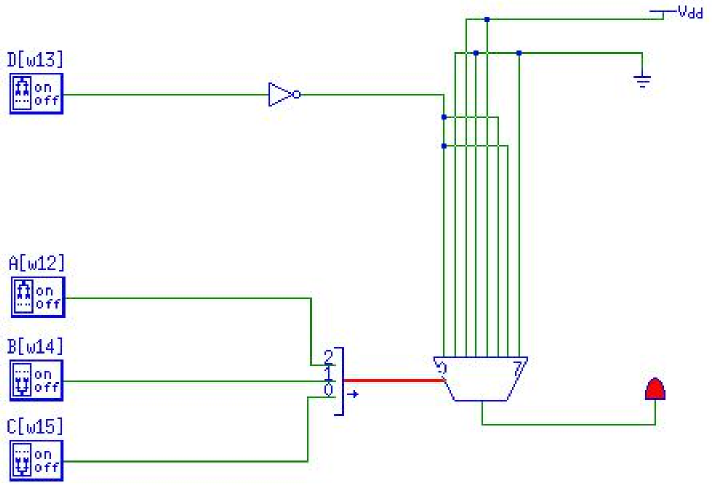
A	*B*	*C*	*D*	*f(A,B,C,D)*
0	1	0	1	1



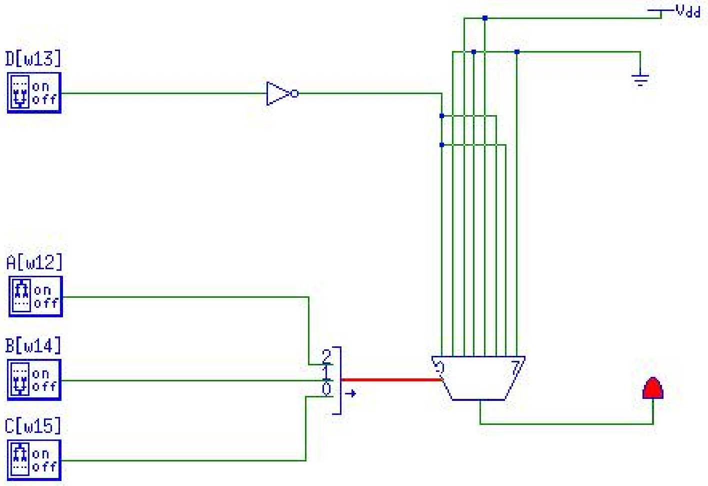
A	*B*	*C*	*D*	*f(A,B,C,D)*
1	0	0	0	1



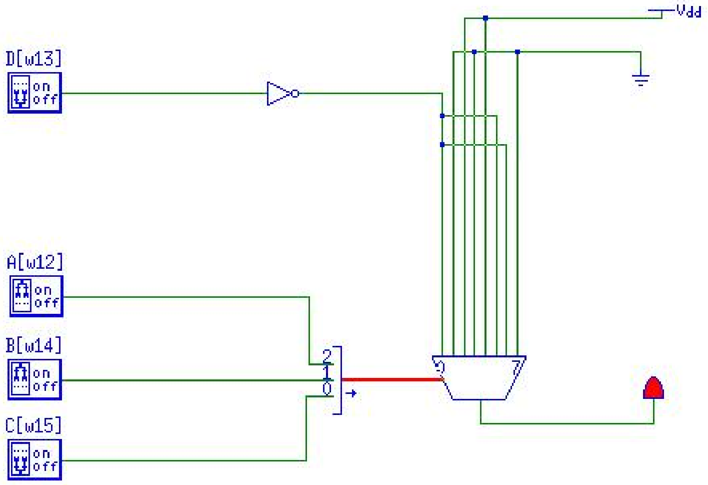
A	*B*	*C*	*D*	*f(A,B,C,D)*
1	0	0	1	1



A	*B*	*C*	*D*	*f(A,B,C,D)*
1	0	1	0	1



A	*B*	*C*	*D*	*f(A,B,C,D)*
1	1	0	0	1


Vdd
D[w13]
on
off
A[w12]
on
off
B[w14]
on
off
C[w15]
on
off
2
1
0
0
7