Name	Muhammad Asad
Reg. #	2019-EE-383
Marks	

Experiment # 8

Designing and investigation of Multiplexers / Demultiplexers

Objective:

- *To understand Multiplexers / Demultiplexers design and function.*
- *Combination of multiplexer and demultiplexer*

Apparatus:

Workstation Core 2 Duo, Uni*Tr@inLucas Nulle*, SO4201-9X Multiplexer / Demultiplexer Card, Jumpers wires, Power Supplies.

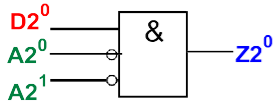
Theory:

- **Multiplexers**

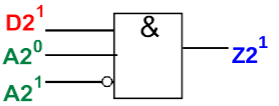
How is the number of data lines (N) related to the number of address lines (n)? Write the expression.

How many address lines would be needed for 8, 16 or 32 data lines?

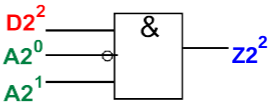
The following represents the logic table for the AND gates in the above illustration.



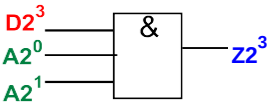
$D2^0$	$A2^1$	$A2^0$	$Z2^0$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0



$D2^0$	$A2^1$	$A2^0$	$Z2^1$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



$D2^0$	$A2^1$	$A2^0$	$Z2^2$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



$D2^0$	$A2^1$	$A2^0$	$Z2^3$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Q. How the process of selecting is implemented in the circuit?

The process is implemented on the base of the switch and inputs. As the switch is given low input (0) logic as a result the circuit off (testing lamp off), and then if the switch give us high input (1) logic circuit will work and on the base selection lines output is produced. The selection is implemented on the base of switches and inputs and it produce outputs.

Demultiplexers

How can the design of the demultiplexer be derived from that of the multiplexer?

The demultiplexer take one single input data line and then switches it to any one of a number of individual output lines one at a time.Is the addressing logic for the demultiplexer different from that of the multiplexer. The data distributor, known as Demultiplexer or "Demux" for short, is the exact opposite of the Multiplexer.

Is the addressing logic for the demultiplexer different from that of the multiplexer?

No, the addressing logic for the demultiplexer is not different from that of the multiplexer the output of the multiplexer and de-multiplexer must be depends upon the selection of addressing lines.

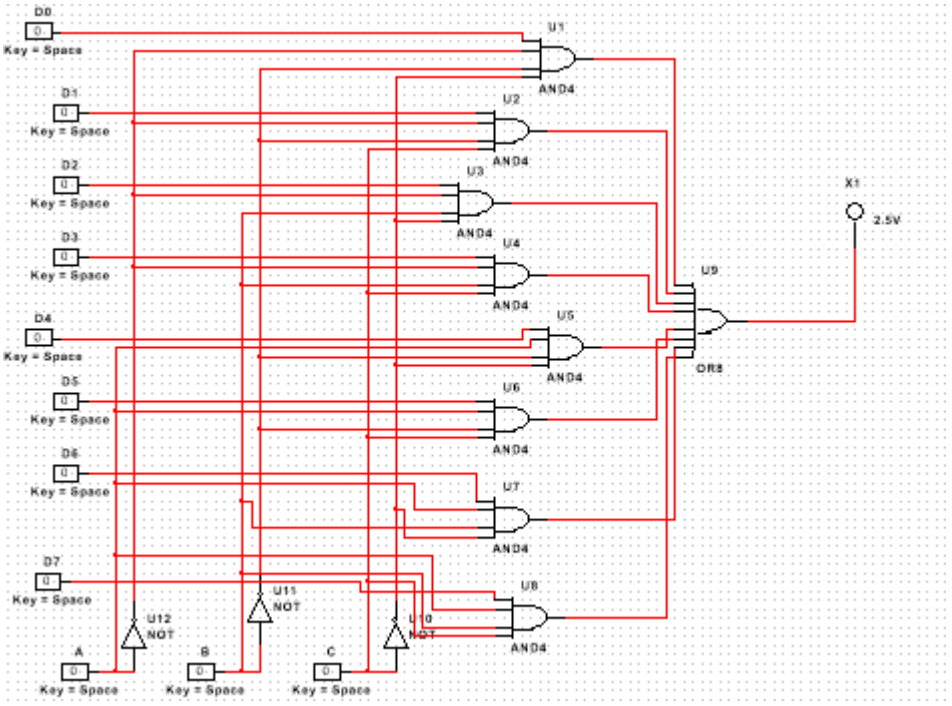
Procedure and observation:

Exercise 1:

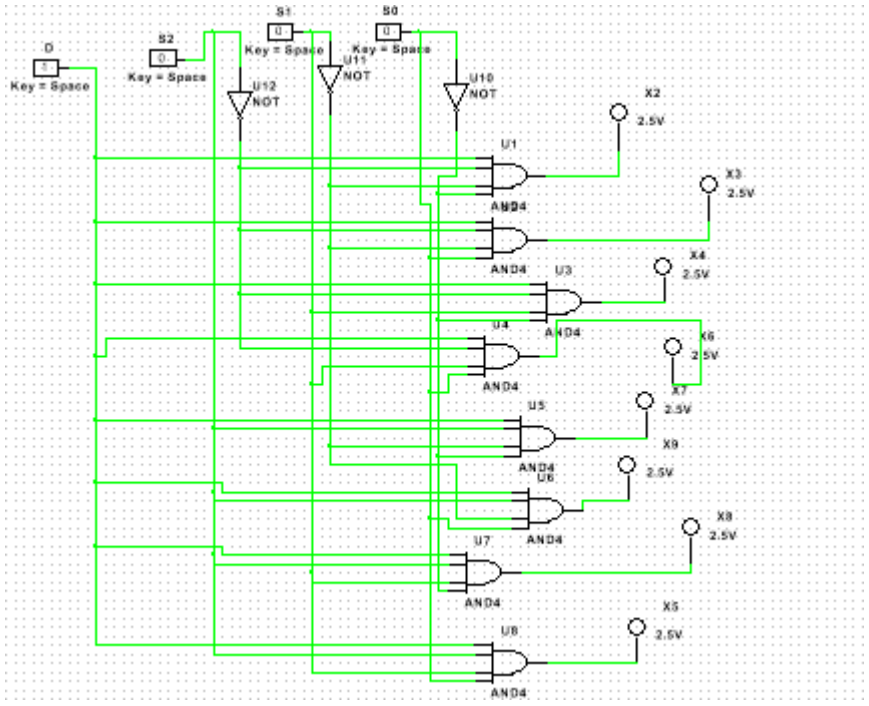
Design an 8×1 Multiplexer and Demultiplexer in HDL on ModelSim.

Circuit Diagram:

Multiplexer Diagram:



De-multiplexer Circuit:



HDL code:

Multiplexer:

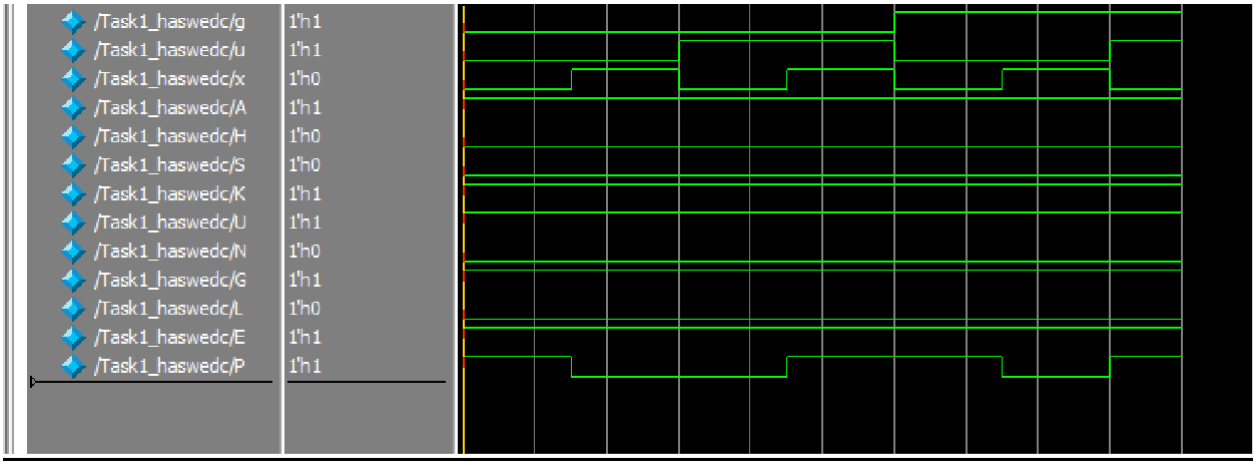
```
module Task1(g,u,x,A,H,S,K,U,N,G,L,E,P);
input g,u,x,A,H,S,K,U,N,G,L,E;
output P;
wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11;
not n1(w1,g);
not n2(w2,u);
not n3(w3,x);
and a1(w4,E,A,w1,w2,w3);
and a2(w5,E,H,w1,w2,x);
and a3(w6,E,S,w1,u,w3);
and a4(w7,E,K,w1,u,x);
and a5(w8,E,U,g,w2,w3);
and a6(w9,E,N,g,w2,x);
and a7(w10,E,G,g,u,w3);
and a8(w11,E,L,g,u,x);
or a9(P,w4,w5,w6,w7,w8,w9,w10,w11);
endmodule
module Task1_haswedc();
reg g,u,x,A,H,S,K,U,N,G,L,E;
wire P;
Task1 t1(g,u,x,A,H,S,K,U,N,G,L,E,P);
initial
begin
g=0; u=0; x=0; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=0; u=0; x=1; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=0; u=1; x=0; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=0; u=1; x=1; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=1; u=0; x=0; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=1; u=0; x=1; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=1; u=1; x=0; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
g=1; u=1; x=1; A=1; H=0; S=0; K=1; U=1; N=0; G=1; L=0;E=1; #15;
```

*De-multi*plexer:

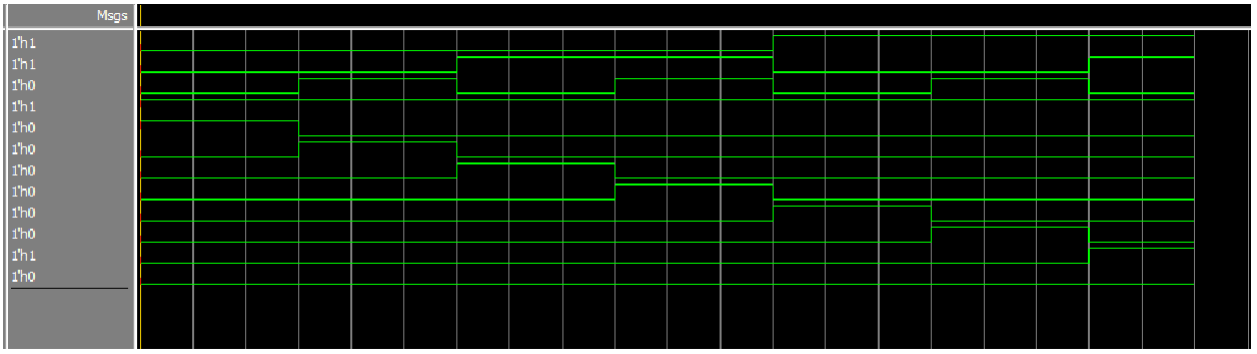
```
module Task2(g,u,x,A,H,S,M,U,N,G,L,T);
input g,u,x,T;
output A,H,S,M,U,N,G,L;
wire w1,w2,w3;
not n1(w1,g);
not n2(w2,u);
not n3(w3,x);
and a1(A,T,w1,w2,w3);
and a2(H,T,w1,w2,x);
and a3(S,T,w1,u,w3);
and a4(M,T,w1,u,x);
and a5(U,T,g,w2,w3);
and a6(N,T,g,w2,x);
and a7(G,T,g,u,w3);
and a8(L,T,g,u,x);
endmodule
module Task_h();
reg g,u,x,T;
wire A,H,S,M,U,N,G,L;
Task2 LP(g,u,x,A,H,S,M,U,N,G,L,T);
initial
begin
g=0; u=0; x=0; T=1; #15;
g=0; u=0; x=1; T=1; #15;
g=0; u=1; x=0; T=1; #15;
g=0; u=1; x=1; T=1; #15;
g=1; u=0; x=0; T=1; #15;
g=1; u=0; x=1; T=1; #15;
g=1; u=1; x=0; T=1; #15;
g=1; u=1; x=1; T=1; #15;
end
endmodule
```

Output waveforms for all combinations:

Multiplexer:

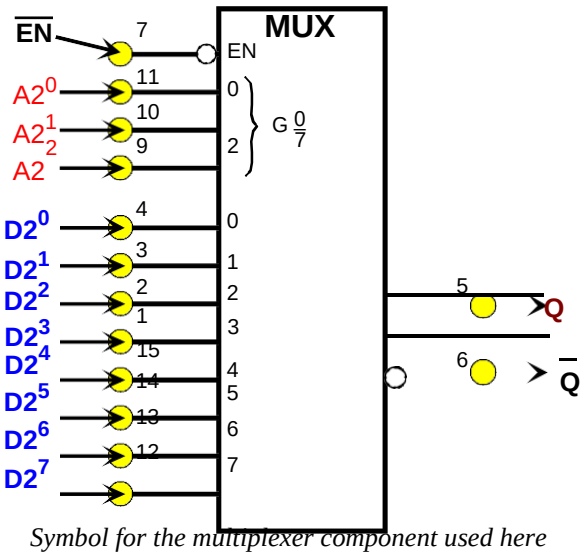


*De-multi*plexer:



Multiplexer – Hardware Trainer Experiments

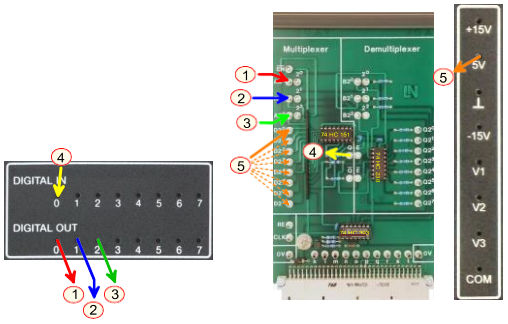
This experiment investigates the basic behavior of a **multiplexer**.



Symbol for the multiplexer component used here with its accessible ports.

Experiment steps: Multiplexer

1. Connect the terminals on the card with those of the Uni*Tr@in*-I as shown in the list of connections to the right.

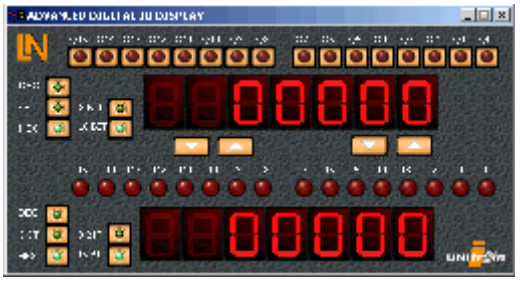


Click on the image for a view of the wiring.

List of connections

From	*To*
Digital out 0	Terminal **A2⁰**
Digital out 1	Terminal **A2¹**
Digital out 2	Terminal **A2²**
Digital in 0	Terminal **Q**

2. Open the following virtual instrument from the *Instruments* menu - Advanced digital IO

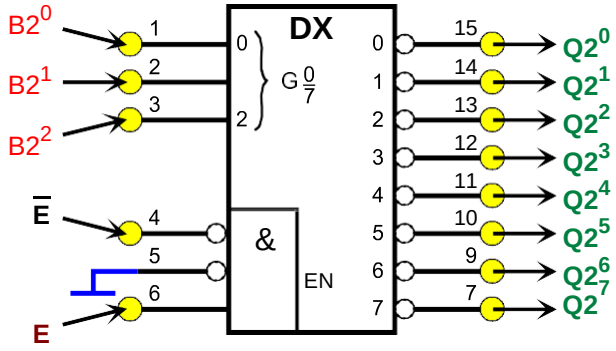


3. Set the input lines ($D2^0$ to $D2^7$) to the logic levels given in the table (high level is achieved by connecting to the 5V-socket of the Experimenter / open sockets are at low level). Set the suggested bit combinations for the address lines **$A2^0$..$A2^2$** in sequence (go through the BCD codes for 0 to 7) and enter the value of **Q** that emerges in each case.

									0	***1***	***2***	***3***	***4***	***5***	***6***	***7***
								$A2^0$	0	1	0	1	0	1	0	1
								$A2^1$	0	0	1	1	0	0	1	1
								$A2^2$	0	0	0	0	1	1	1	1
$D2^7$	**$D2^6$**	**$D2^5$**	**$D2^4$**	**$D2^3$**	**$D2^2$**	**$D2^1$**	**$D2^0$**		**Q**							
0	0	0	0	0	0	0	0		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	0	0	0	0	**1**	0		0	1	0	0	0	0	0	0
0	0	0	0	0	**1**	0	0		0	0	1	0	0	0	0	0
0	0	0	0	**1**	0	0	0		0	0	0	1	0	0	0	0
0	0	0	**1**	0	0	0	0		0	0	0	0	1	0	0	0
0	0	**1**	0	0	0	0	0		0	0	0	0	0	1	0	0
0	**1**	0	0	0	0	0	0		0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0		0	0	0	0	0	0	0	1

Demultiplexer - Experiments

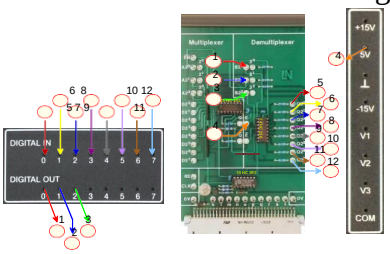
This experiment studies the basic behavior of a demultiplexer.



Symbol for the multiplexer component used here with its accessible ports.

Experiment steps: Demultiplexer

1. Connect the terminals on the card with those of the Uni*Tr@in*-I as shown in the list of connections to the right.

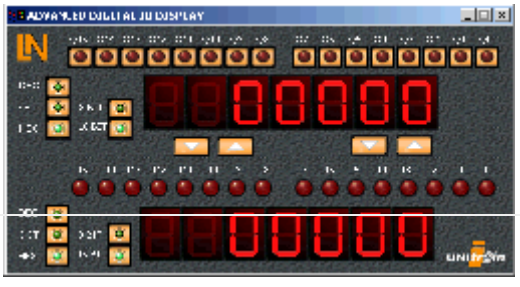


Click on the image for a view of the wiring.

List of connections

From	*To*
Digital out 0	Terminal **B2⁰**
Digital out 1	Terminal **B2¹**
Digital out 2	Terminal **B2²**
Experimenter **5V**	Terminal **E**
Digital in 0	Terminal **Q2⁰**
Digital in 1	Terminal **Q2¹**
Digital in 2	Terminal **Q2²**
Digital in 3	Terminal **Q2³**
Digital in 4	Terminal **Q2⁴**
Digital in 5	Terminal **Q2⁵**
Digital in 6	Terminal **Q2⁶**
Digital in 7	Terminal **Q2⁷**

2. Open the following virtual instrument from the *Instruments* menu - Advanced digital IO



3. Connect socket "**E**" to logical "**1**", (the 5V output of the Experimenter). Set the specified bit patterns for the address lines $B2^0..B2^2$ in sequence (go through the BCD codes for 0 to 7) and enter the values of $Q2^0 .. Q2^7$ that emerge in each case.

	$B2^2$	$B2^1$	$B2^0$	$Q2^7$	$Q2^6$	$Q2^5$	$Q2^4$	$Q2^3$	$Q2^2$	$Q2^1$	$Q2^0$
0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	1	1	0	1
2	0	1	0	1	1	1	1	1	0	1	1
3	0	1	1	1	1	1	1	0	1	1	1
4	1	0	0	1	1	1	0	1	1	1	1
5	1	0	1	1	1	0	1	1	1	1	1
6	1	1	0	1	0	1	1	1	1	1	1
7	1	1	1	0	1	1	1	1	1	1	1

4. Connect input $\overline{E}$ to logical "**0**" (one of the ground connections on the card). Repeat the above experiment.

   What do you observe?

   As we select the input E' as a logic 0 and 1, because input E' invert the value and gate and as result it produce 1.And then the input logic 000 then the according to the table except 1st output all are 1,And when the input is 001 then the 2nd output is zero and so on..

5. Then connect "**E'**" to the **5V** socket (logical "**1**") and repeat the experiment again.

   What do you observe?

   As we select E' to the 5V socket (logical 1), As we know E' invert the value so that's why all the AND gate output 0 (low value) and the output of this also inverts the value so as a result all probes will give us logical 1 (high value)..