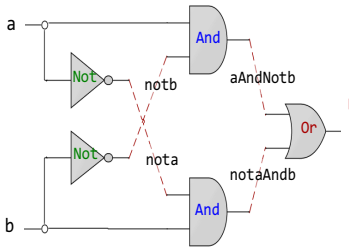
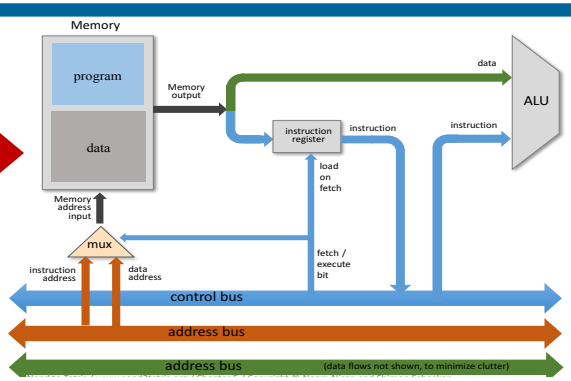
Computer Organization & Assembly Language Programming




```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
  Not(in=a, out=nota);
  Not(in=b, out=notb);
  And(a=nota, b=b, out=w1);
  And(a=a, b=notb, out=w2);
  Or(a=w1, b=w2, out=out);
}
```




```
@R1
D=M
@temp
M=D
```

```
0000000000000001
1111110000010000
0000000000010000
1110001100001000
```

Lecture # 04

HDL for Combinational Circuits - III

```
#include<stdio.h>
#include<stdlib.h>
int main(){
  printf("Learning is fun with Arif\n");
  exit(0);
}
```

```
global main
SECTION .data
   msg: db "Learning is fun with Arif", 0Ah, 0h
   len_msg: equ $ - msg
SECTION .text
   main:
      mov rax,1
      mov rdi,1
      mov rsi,msg
      mov rdx,len_msg
      syscall
      mov rax,60
      mov rdi,0
      syscall
```

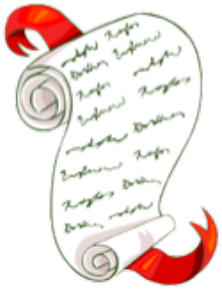
```
0:  b8 01 00 00 00
5:  bf 01 00 00 00
a:  48 be 00 00 00 00 00
11: 00 00 00
14: ba 1b 00 00 00
19: 0f 05
1b: b8 3c 00 00 00
20: bf 00 00 00 00
25: 0f 05
```

Slides of first half of the course are adapted from:
https://www.nand2tetris.org
Download s/w tools required for first half of the course from the following link:
https://drive.google.com/file/d/0B9c0BdDJz6XpZUh3X2dPR1o0MUE/view

Instructor: Muhammad Arif Butt, Ph.D.

Today's Agenda

- Class Quiz
- Review of Script Based Simulation with output and compare files
- Writing HDL for Combinational Circuits like
  - Decoder
  - Encoder
  - Multiplexer
  - De-Multiplexer
- Demo on above chips on H/W Simulator



Class Quiz

- The Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, …) is an infinite sequence, in which the first two terms are 1 and 1, and each succeeding term is the sum of the two immediately proceeding terms
- Design a combinational circuit that receives a four bit binary number as input. The output generated by the circuit is 1, if the input number is a Fibonacci number, otherwise the output is zero
- Also write down the HDL for this chip and verify chip logic by loading it in the h/w simulator. While writing the HDL you can assume that And, Not and Or chips are available to you

chip interface

Fibo.hdl

```
/** Check Fibonacci number */
CHIP Fibo {
  IN w, x, y, z;
  OUT out;
  PARTS:
   //Write chip implementation code
}
```

Review Script-Based Simulation

Xor.hdl

```
CHIP Xor {
  IN a, b;
  OUT out;
  PARTS:
    Not(in=a, out=nota);
    Not(in=b, out=notb);
    And(a=a, b=notb, out=aAndNotb);
    And(a=nota, b=b, out=notaAndb);
    Or(a=aAndNotb, b=notaAndb, out=out);
```

Tested chip

Xor.tst

```
Load Xor.hdl,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;
set a 0, set b 0, eval, output;
set a 0, set b 1, eval, output;
set a 1, set b 0, eval, output;
set a 1, set b 1, eval, output;
```

test script

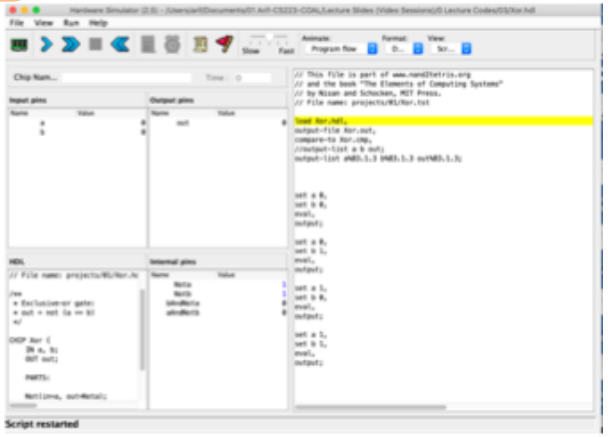
output File, created by the test script as a side-effect of the simulation process

Xor.out

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

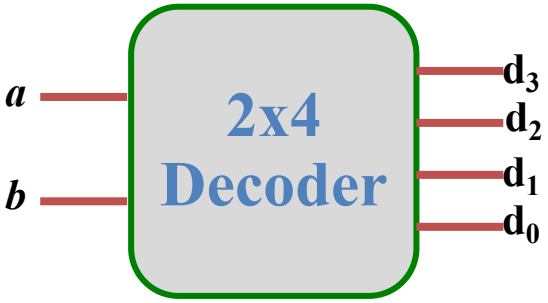
Xor.cmp

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0



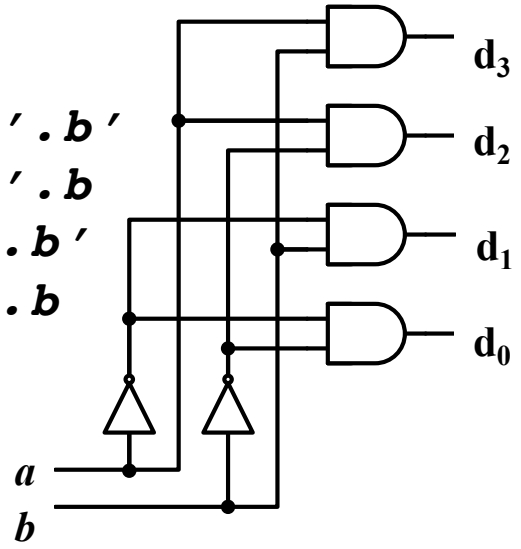
Review of Combinational Circuits

2x4 Decoder



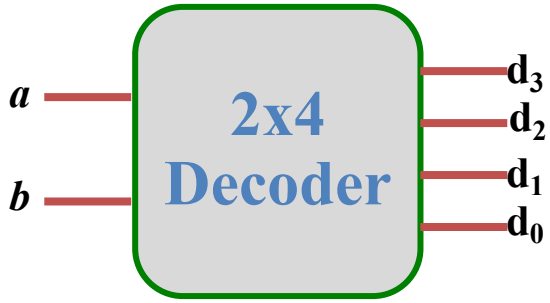
a	b	d_0	d_1	d_2	d_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

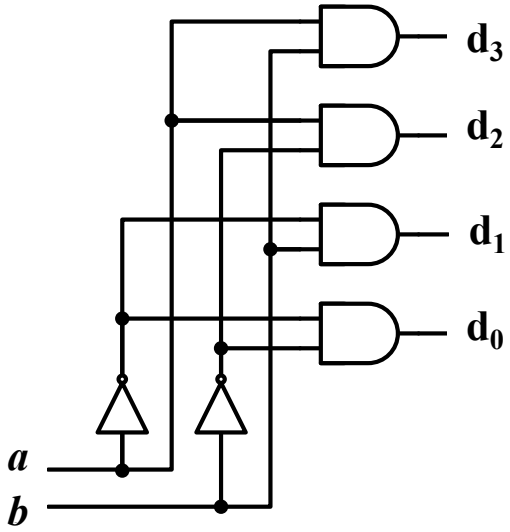
$d_0 = a'.b'$
$d_1 = a'.b$
$d_2 = a.b'$
$d_3 = a.b$



- A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines
- If a binary decoder receives n inputs it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated
- If the n-bit coded information at the input has unused combinations, the decoder may have less than 2^n output lines, e.g., a BCD-to-7 segment decoder is actually a 4x16 decoder, in which the output lines from d_{10} to d_{15} are not used
- A gate level diagram of a 2x4 decoder is shown above

2x4 Decoder Implementation



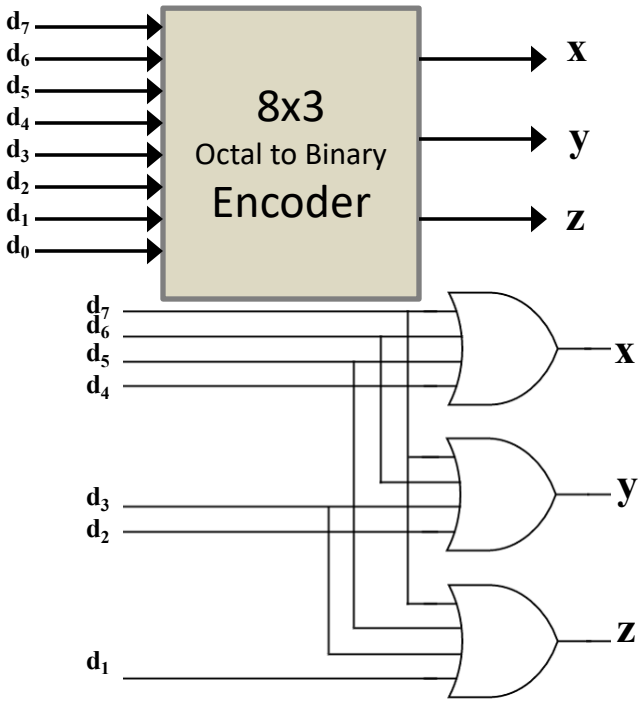


Decoder.hdl

```
CHIP Decoder {
 IN a, b;
 OUT d0, d1, d2, d3;
 PARTS:
 Not(in=a, out=Nota);
 Not(in=b, out=Notb);
 And(a=Nota, b=Notb, out=d0);
 And(a=Nota, b=b, out=d1);
 And(a=a, b=Notb, out=d2);
 And(a=a, b=b, out=d3);
}
```

$d_0 = a'.b'$
$d_1 = a'.b$
$d_2 = a.b'$
$d_3 = a.b$

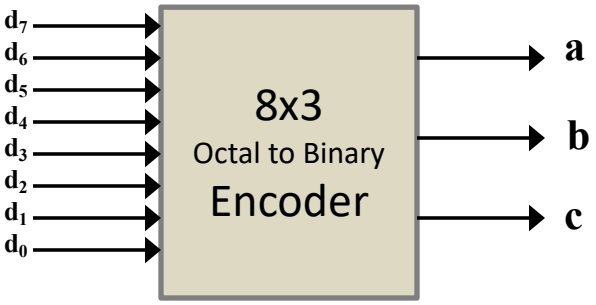
8x3 Encoder

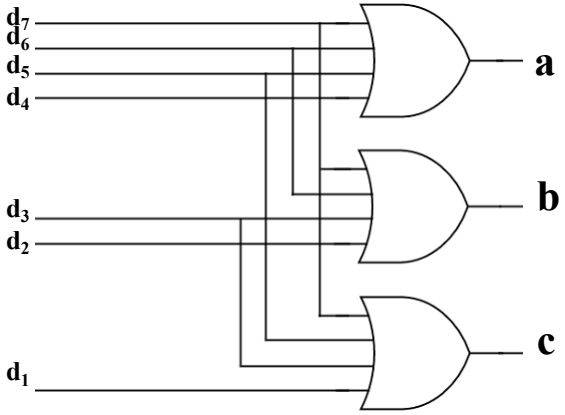


d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	x	y	z
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- An encoder is a combinational circuit that converts binary information in the form of a 2^n input lines into n output lines, which represent n-bit code for the input. Normally used to encode various symbols and alphabetic characters to a coded format
- For example: Octal to Binary encoder (shown above) takes 8 input lines and generates 3 output lines. Similarly, an ASCII encoder will have 128 input lines and 7 output lines
- **Limitations**:
  - Output 0 may mean that all inputs are zero or may mean that d_0 is one
  - If more than one input lines are high the encoder generates unpredictable output

8x3 Encoder Implementation





$$a = d_4 + d_5 + d_6 + d_7$$
$$b = d_2 + d_3 + d_6 + d_7$$
$$c = d_1 + d_3 + d_5 + d_7$$

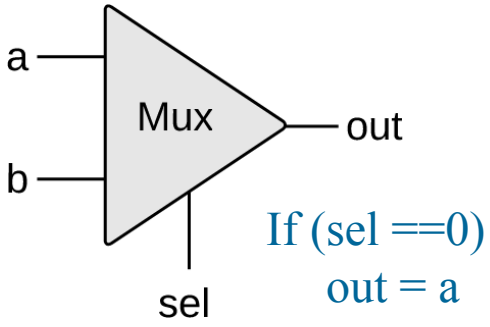
Encoder.hdl

```
CHIP Encoder {
 IN d0,d1,d2,d3,d4,d5,d6,d7;
 OUT a, b, c;
 PARTS:
   Or(a=d4, b=d5, out=w1);
   Or(a=d6, b=d7, out=w2);
   Or(a=w1, b=w2, out=a);

   Or(a=d2, b=d3, out=w3);
   Or(a=d6, b=d7, out=w4);
   Or(a=w3, b=w4, out=b);

   Or(a=d1, b=d3, out=w5);
   Or(a=d5, b=d7, out=w6);
   Or(a=w5, b=w6, out=c);
}
```

2x1 Mux



If (sel ==0)
    out = a
else
    out = b

a	b	sel	out
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**0**
1	1	1	**1**

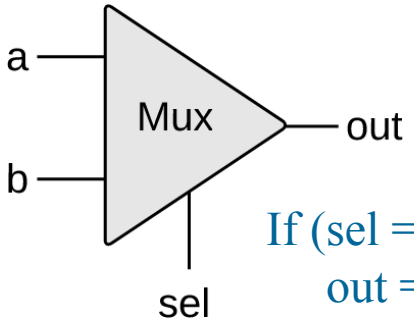
sel out

sel	out
0	**a**
1	**b**

abbreviated truth table

- Multiplexing is a technique of putting multiple signals on a single data path
- A Multiplexer (data selector) is a combinational circuit that selects binary information from one of many (2^n) input lines and transmit it to a single output line. The selection of a particular input line is controlled by a set of n selection lines
- A 2-way multiplexor (2x1 Mux) enables selecting, and outputting, one of two possible inputs as shown above

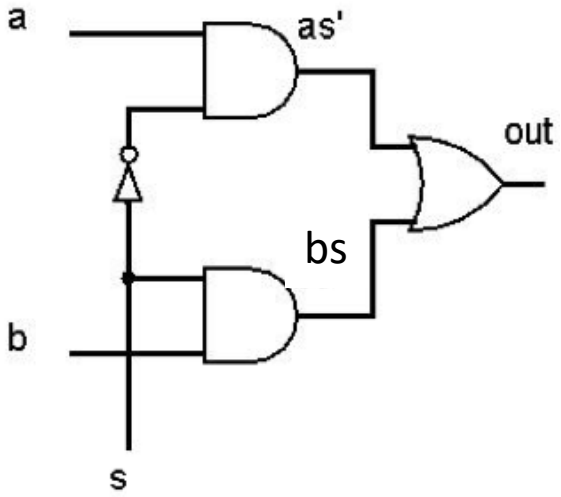
2x1 Mux Implementation



If (sel ==0)
    out = a
else
    out = b

sel	out
0	a
1	b

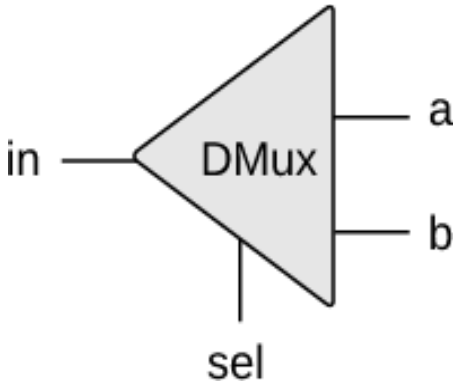
out = aS′ + bS



Mux.hdl

```
CHIP Mux {
 IN a, b, sel;
 OUT out;
 PARTS:
 Not(in=sel, out=Notsel);
 And(a=a, b=Notsel, out=aAndNotsel);
 And(a=b, b=sel, out=bAndsel);
 Or(a=aAndNotsel, b=bAndsel, out=out);
}
```

1x2 Dmux

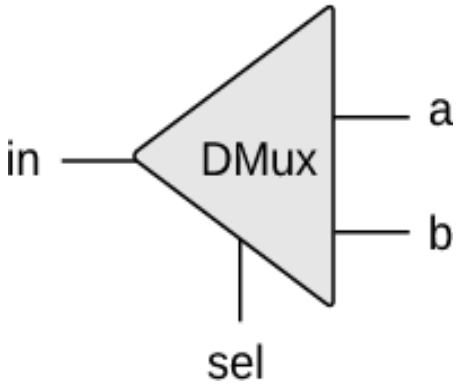



```
If (sel==0)
   {a,b} = {in,0}
else
   {a,b}={0,in}
```

in	sel	a	b
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

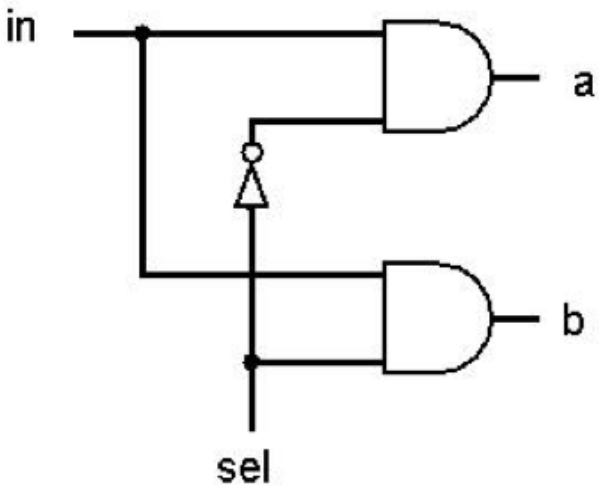
- A DMultiplexer (data distributer) is a combinational circuit that receives information on a single line and transmit this information on one of 2^n possible output lines. The selection of a particular output line is controlled by a set of n selection lines
- A 1x2 dmultiplexor distributes the single input value into one of two possible destinations as shown above

1x2 Dmux Implementation




```
If (sel==0)
   {a,b} = {in,0}
else
   {a,b}={0,in}
```

in	sel	a	b
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1



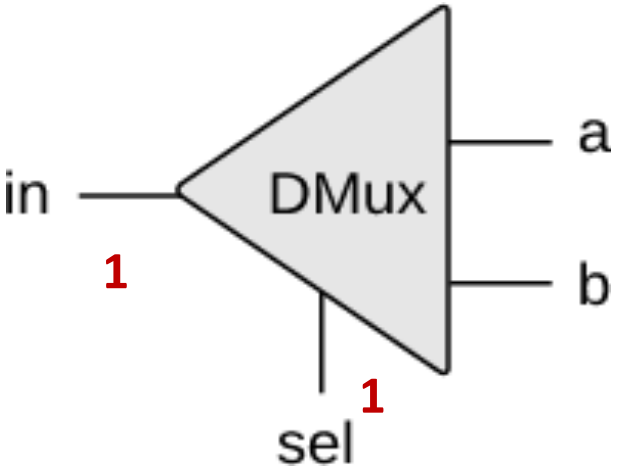
DMux.hdl

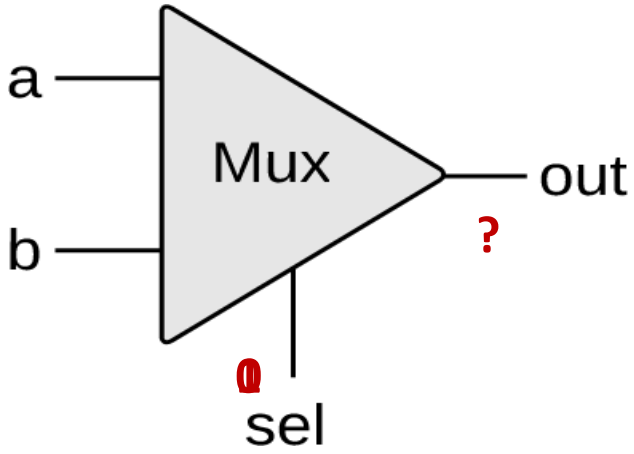
```
CHIP DMux{
   IN in, sel;
   OUT a, b;

   PARTS:
      Not(in=sel, out=Notsel);
      And(a=in, b=Notsel, out=a);
      And(a=in, b=sel, out=b);
}
```

a = in.S'
b = in.S

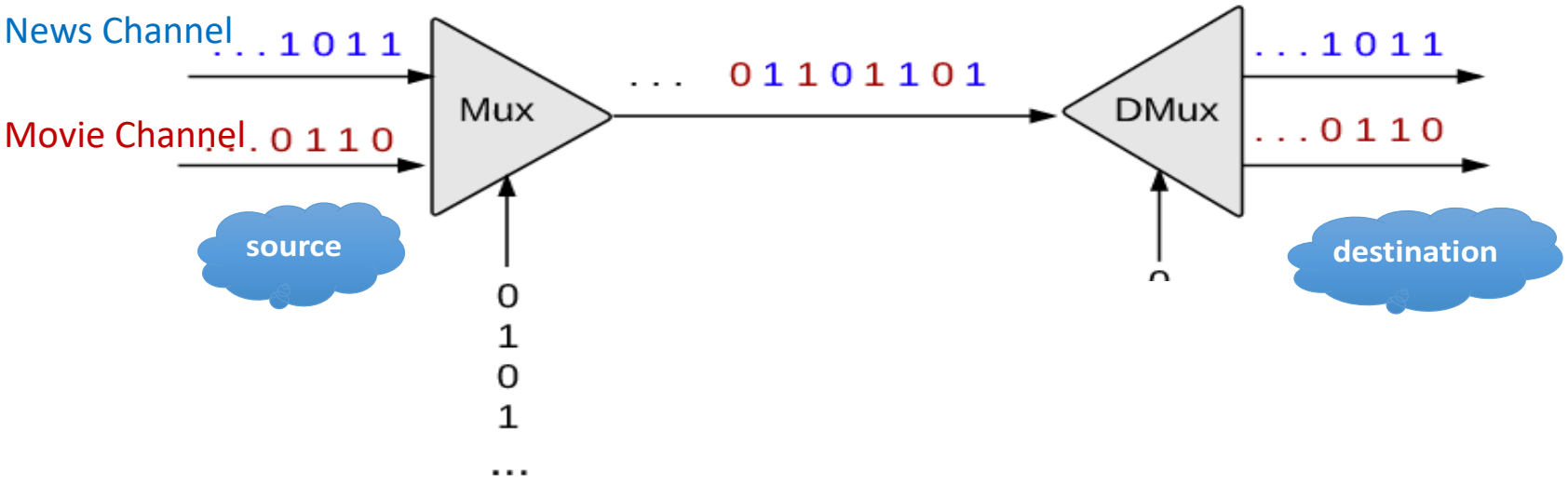
Question





- Consider the above scenario, in which we have a 1x2 DMux and a 2x1 Mux
- The two outputs of Dmux are fed into the two inputs of the Mux
- Suppose the **in** of the Dmux is 1, and its **sel** input is also 1
- What is there on the **out** of Mux, if the **sel** of the Mux is set to 0?
- What is there on the **out** of Mux, if the **sel** of the Mux is set to 1?

Example: Multiplexing/Demultiplexing In Communication Network



- A common use of multiplexing / de-multiplexing logic is shown above that enables transmitting multiple channels/messages on a single shared communication line
- Suppose we have two channels coming in the 2x1 Mux (source). We want to transmit both these channels via a single communication line
- The **sel** bit of the Mux is connected to an oscillator that produces a repetitive train of alternating 0 and 1 signals, so transmitting one bit from each channel in each oscillation (Time Division Multiplexing)
- On the destination side, one can use the **sel** input of the Dmux to select the channel to watch ☺

Demo Combinational Circuits

Things To Do

- Perform interactive testing of the chips designed in today's session on the h/w simulator. You can download the .hdl, .tst and .cmp files of above chips from the course bitbucket repository:

https://bitbucket.org/arifpucit/coal-repo/

- Design different sizes of decoder, encoder, multiplexer, and de-multiplexer circuits. Write their .hdl files and confirm the chip logic by loading the .hdl files in the h/w simulator

- Interested students should also try to implement all the standard logic gates (Not, And, Or, Nand, Nor, Xor, Xnor) with multiplexer(s)

Coming to office hours does NOT mean you are academically week!