Digital Systems I

Lab06-07 Slides based on **Chapter 5 - Part II**

Behavioral Modeling of Digital Circuits

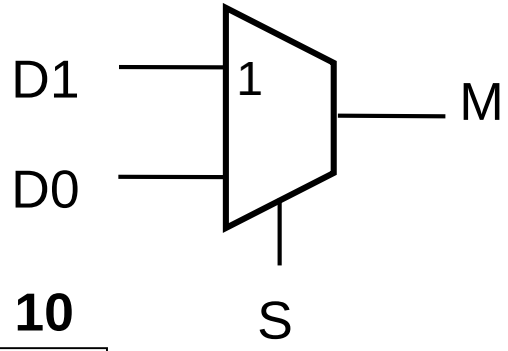
Conditional Signal Assignments

and

Selected Signal Assignments

N. Tabrizi Kettering University

Row	D1	D0	S	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

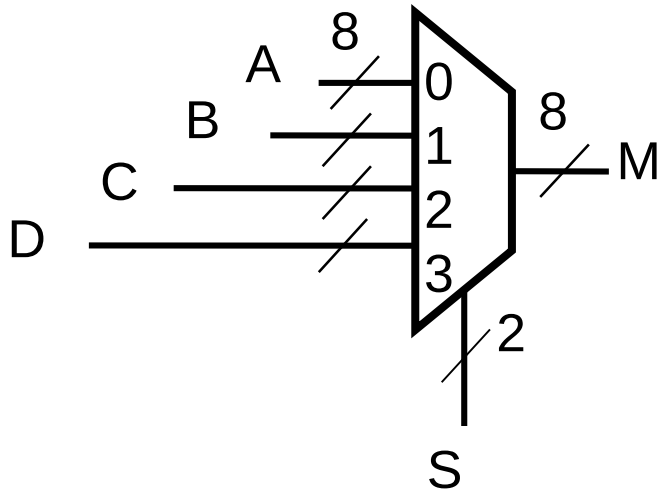


S \ D1 D0	00	01	11	10
0	0	1	1	0
1	0	0	1	1

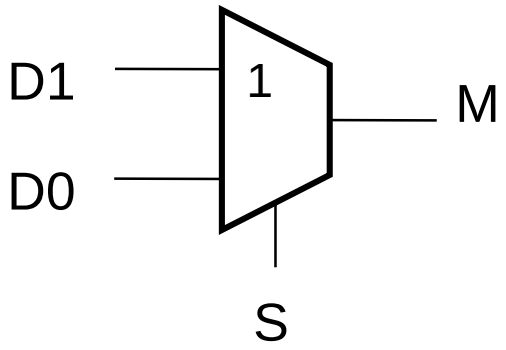
M = S . D1 + S' . D0

M <= (NOT S AND D0) OR (S AND D1);

Simple Signal Assignment ☺

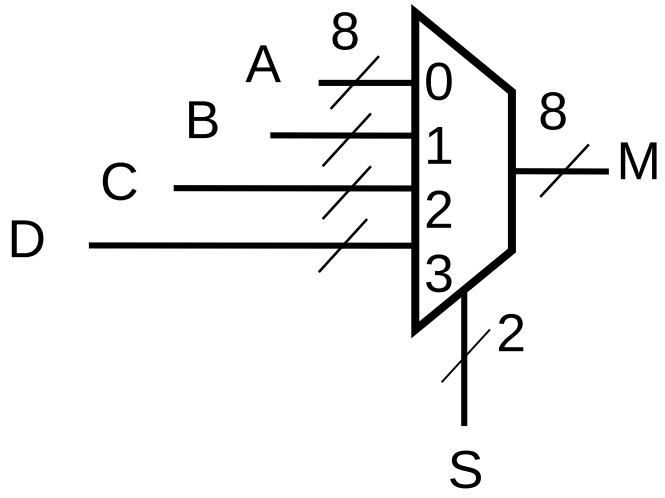


Simple Signal Assignment ☹



M <= D1 WHEN S = ‘1’ ELSE D0;

Conditional Signal Assignment



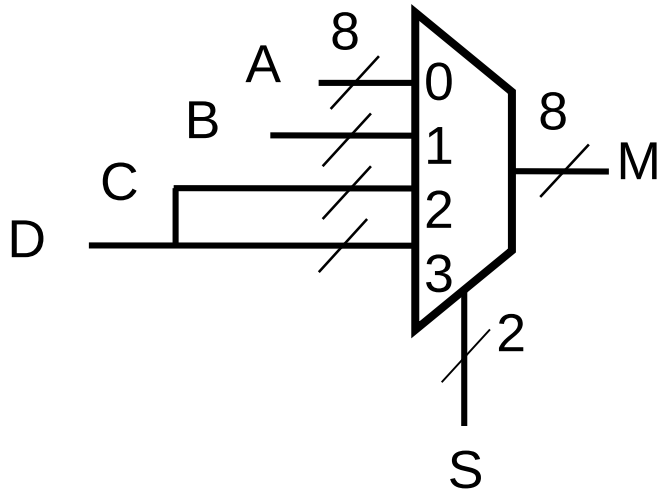
In this specific example,

Order is unimportant

```
M <=    A WHEN  S = “00”  ELSE
        B WHEN  S = “01”  ELSE
        C WHEN  S = “10”  ELSE
        D;
```

-- Use “ ” for vectors

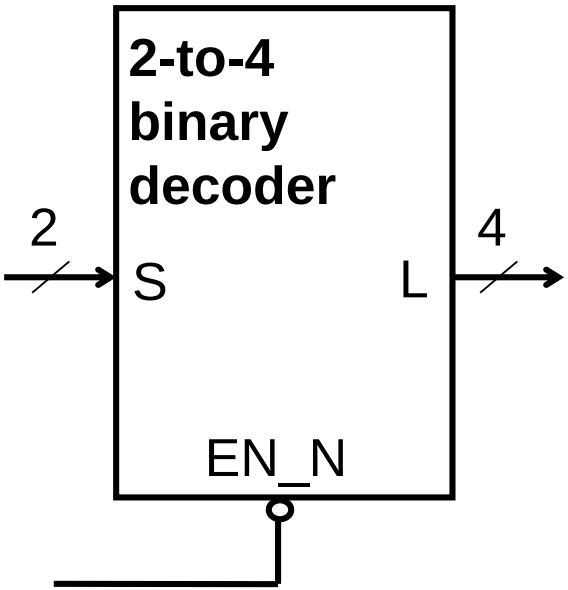
Conditional Signal Assignment




```
M <=    A WHEN  S = “00”  ELSE          -- Use “ ” for vectors
        B WHEN  S = “01”  ELSE
        D;
```

Conditional Signal Assignment

Binary Decoders



EN_N	S	L
0	00	0001
0	01	0010
0	10	0100
0	11	1000
1	xx	0000

Binary Decoders

Now

order is important!

```
L <=    “0000”  WHEN  EN_N  = ‘1’    ELSE
        “0001“  WHEN    S   = “00”   ELSE
        “0010”  WHEN    S   = “01”   ELSE
        “0100”  WHEN    S   =“10”    ELSE
        “1000” ;
```

EN_N	S	L
0	00	0001
0	01	0010
0	10	0100
0	11	1000
1	xx	0000

Lab06 Lecture Exercise

Look at the following VHDL code:

```
L <=   “1111”WHEN      EN    = ‘0’    ELSE
       “1110“WHEN      S =   “00”     ELSE
       “1101” WHEN     S =   “01”     ELSE
       “1011”WHEN      S =   “10”     ELSE
       “0111” ;
```

Which frequently used digital circuit is described by this code?

Are the outputs active-high or active-low?

active-high active-low

Is the Enable input (EN) active-high or active-low?

active-high active-low

We get this if the first two lines are swapped:

```
L <=   “1110“        WHEN S =   “00”   ELSE
       “1111”        WHEN EN =  ‘0’    ELSE
       “1101”        WHEN S =   “01”   ELSE
       “1011”        WHEN S =   “10”   ELSE
       “0111” ;
```

Give a counterexample to show that the above two codes are not equivalent: (fill in the blanks)

EN = S = L in 1st code = L in 2nd code =

Fill in the blank:

In this example, the order of WHEN ELSE lines is …

Important Unimportant

Write an input value that generates the same output for both codes:

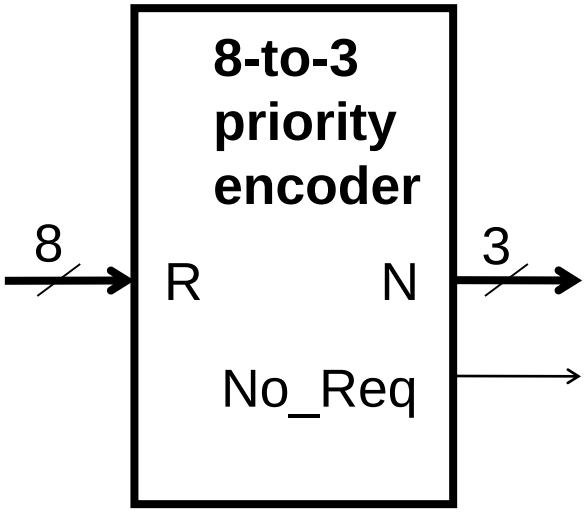
EN = S =

If the last 2 WHEN ELSE lines are swapped, will the meaning of the code change?

Yes No

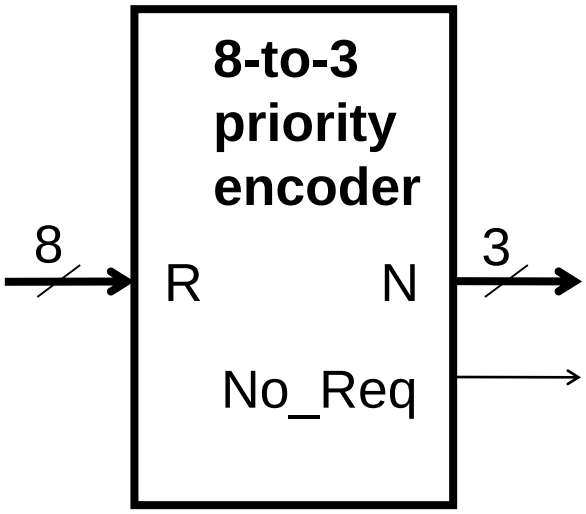
Priority Encoders

7: Highest



R0	R1	R2	R3	R4	R5	R6	R7	N2	N1	N0	No Req
x	x	x	x	x	x	x	1	1	1	1	0
x	x	x	x	x	x	1	0	1	1	0	0
x	x	x	x	x	1	0	0	1	0	1	0
x	x	x	x	1	0	0	0	1	0	0	0
x	**x**	**x**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
x	x	1	0	0	0	0	0	0	1	0	0
x	1	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	0	0	0	0	0	x	x	x	1

Priority Encoders




```
N <=   "111"  WHEN  R(7)  = '1'   ELSE -- Highest
       "110"  WHEN  R(6)  = '1'   ELSE
       "101"  WHEN  R(5)  = '1'   ELSE
       "100"  WHEN  R(4)  = '1'   ELSE
       "011"  WHEN  R(3)  = '1'   ELSE
       "010"  WHEN  R(2)  = '1'   ELSE
       "001"  WHEN  R(1)  = '1'   ELSE
       "000" ;
```

--In this example,

Order is Important

```
--If NO request:
NO_REQ <= '1' WHEN R = "00000000" ELSE '0';
```

Concatenation operator:

N <= D (0) **&** D (3 DOWNTO 1);

Example:
If D = **1010**
then: D (0) = **0**, D (3 DOWNTO 1) = **101**
Therefore: N = **0** &**101** = **0101**

***

M <= K (4 DOWNTO 2) **&** L (3 DOWNTO 1);

Example:
If K = 100101 and L = 0010001, then
M = 001100

Comparators

```
AGTB <= '1'  WHEN  A > B   ELSE    '0';
```

Other VHDL *relational* operators are as follows:

- Equal =
- Not equal /=
- Less than <
- Less than or equal to <=
- Greater than >
- Greater than or equal to >=

```
AGTB <= '1'  WHEN  A > B                          ELSE    '0';
```

Logical and Relational operators may be combined:

```
GT   <= '1'  WHEN  A1 > B1 AND B2 < A2     ELSE    '0';

y    <= '1'  WHEN  (a AND NOT b) = '1'     ELSE    '0';

z    <= '1'  WHEN  a = '1' AND b = '0'     ELSE    '0';
```

```
y   <= '1'  WHEN  ((a AND NOT b) = '1') AND (A1 < B1) ELSE    '0';
```

Lab06 (Digitals I)

Behavioral Modeling of Digital Circuits

Conditional Signal Assignments

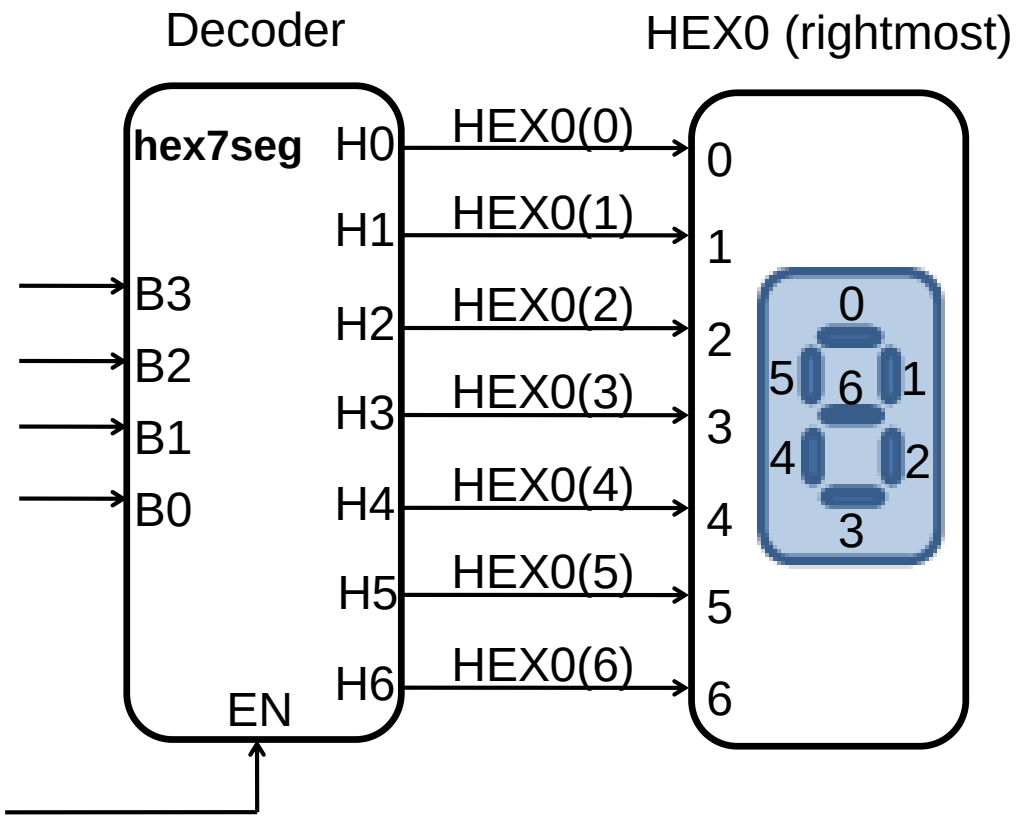
1. Two Textbooks and 8 Students

2. Min_Max

Objectives

- More frequently used digital circuits
- Conditional Signal Assignments in VHDL

Hex to 7-segment Decoder and 7-segment Display



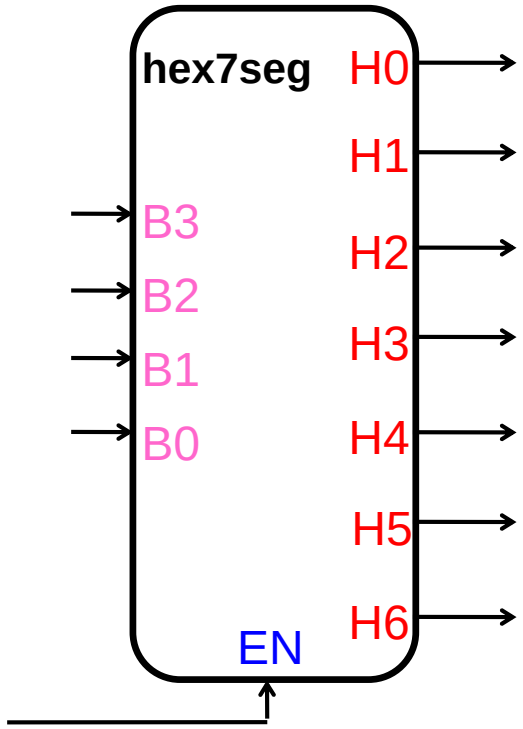
B and H are vectors.

Five more 7-segment displays on the DE1-Lite board:

HEX5 (leftmost), HEX4, HEX3, HEX2, HEX1

Package: (a .vhd file)

Holds component declarations:



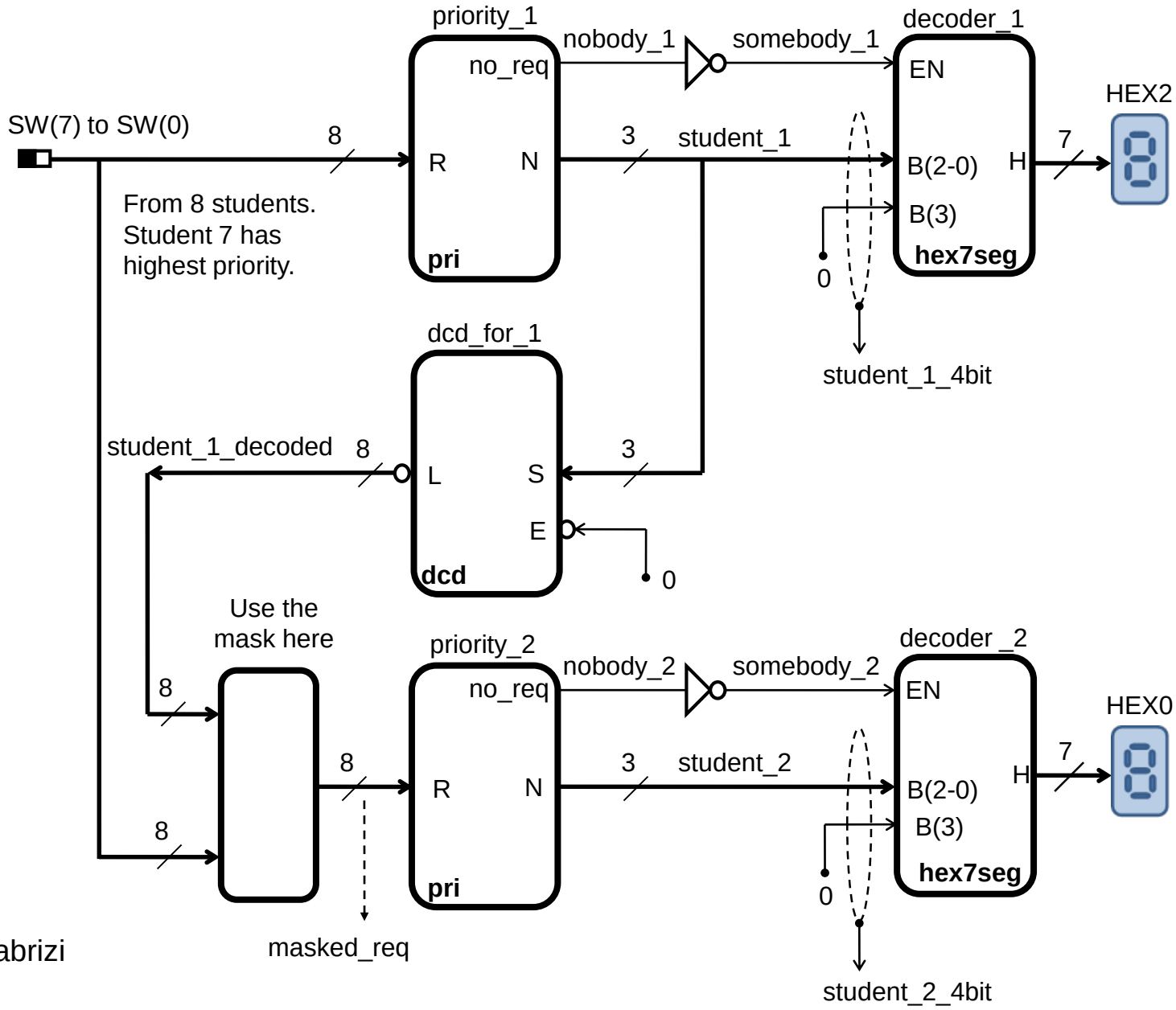
Example:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
PACKAGE dig1pack IS

COMPONENT hex7seg
PORT (   B      : IN     STD_LOGIC_VECTOR (3 DOWNTO 0);
        EN      : IN     STD_LOGIC;    -- Active-high enable input.
         H      : OUT    STD_LOGIC_VECTOR (6 DOWNTO 0)
        );
END COMPONENT;

END dig1pack;
```

Put this line in any file that uses the package:

USE work.dig1pack.all;

Two Textbooks and 8 Students



Chapter 5 - Part II

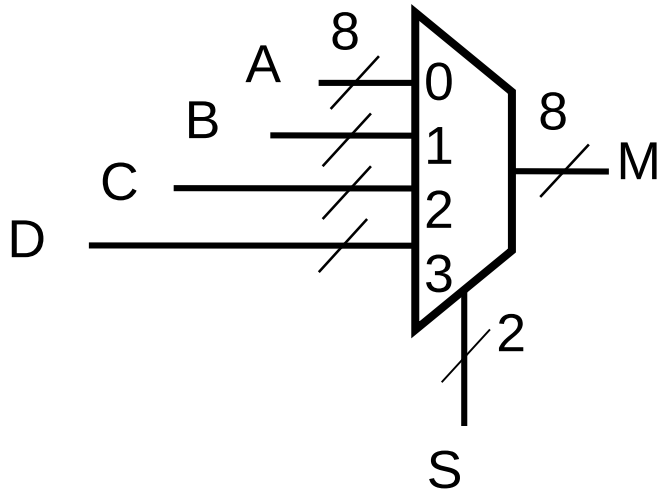
Behavioral Modeling of Digital Circuits

Conditional Signal Assignments

and

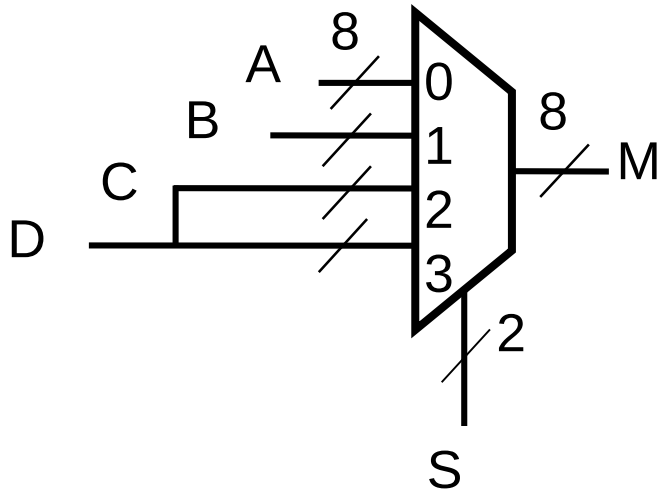
Selected Signal Assignments

N. Tabrizi Kettering University




```
WITH S SELECT
        M <=    A  WHEN “00”,
                B  WHEN  “01”,
                C  WHEN “10”,
                D  WHEN  OTHERS;
```

Selected Signal Assignment




```
WITH S SELECT
        M <=    A  WHEN “00”,
                B  WHEN  “01”,
                D  WHEN  OTHERS;
```

Selected Signal Assignment

Truth Tables

```
WITH ST SELECT

MJF <=      '0'     WHEN  "000",
            '0'     WHEN  "001",
            '0'     WHEN  "010",
            '1'     WHEN  "011",
            '0'     WHEN  "100",
            '1'     WHEN  "101",
            '1'     WHEN  "110",
            '1'     WHEN  OTHERS;
```

Row	ST	MJF
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1

Truth Tables

Shorthand

WITH **ST** SELECT

```
MJF <=  '0'   WHEN "000" | "001" | "010" | "100",
        '1'   WHEN OTHERS;
```

Row	ST	MJF
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1

Lab 07 (Digitals I)

Behavioral Modeling of Digital Circuits

Selected Signal Assignments

Basic Arithmetic and **Logic Unit (ALU)**

Objectives

- Signed/Unsigned Addition/Subtraction/Comparison
- Selected Signal Assignments in VHDL

Arithmetic Logic Unit (ALU)

You design a simple ALU today.

ALU is a basic building block of Microprocessors.

So:

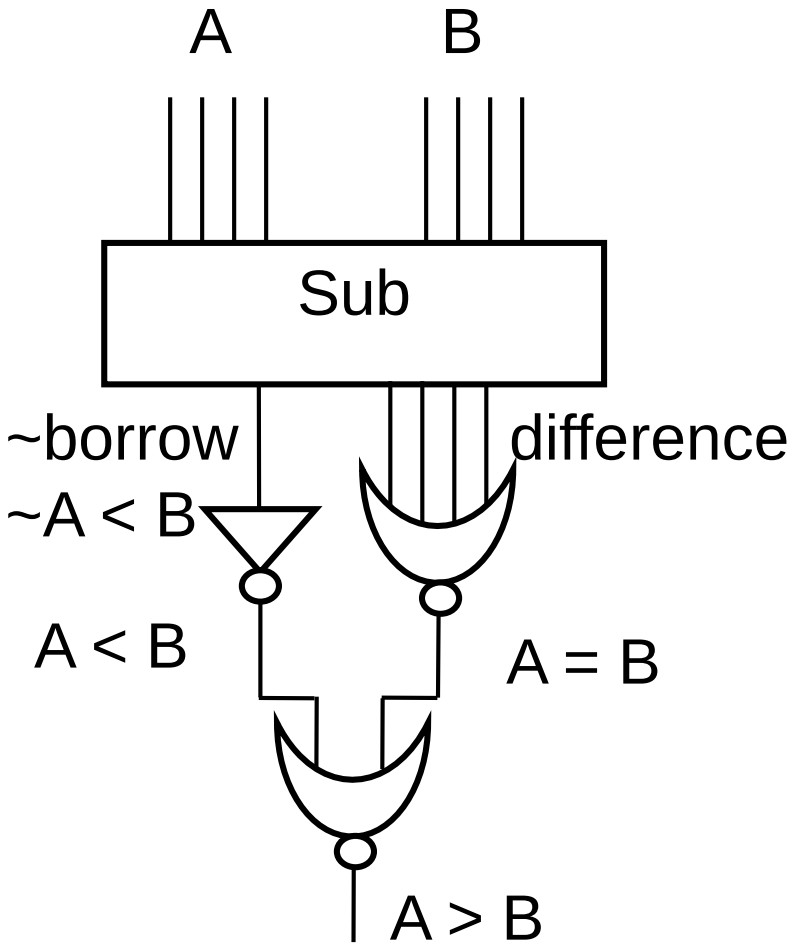
No ALU, No Microprocessor

No Microprocessor, No Computer

No Computer, No …. (Fill in the blank!)

From Lecture Slides, Chapter 5

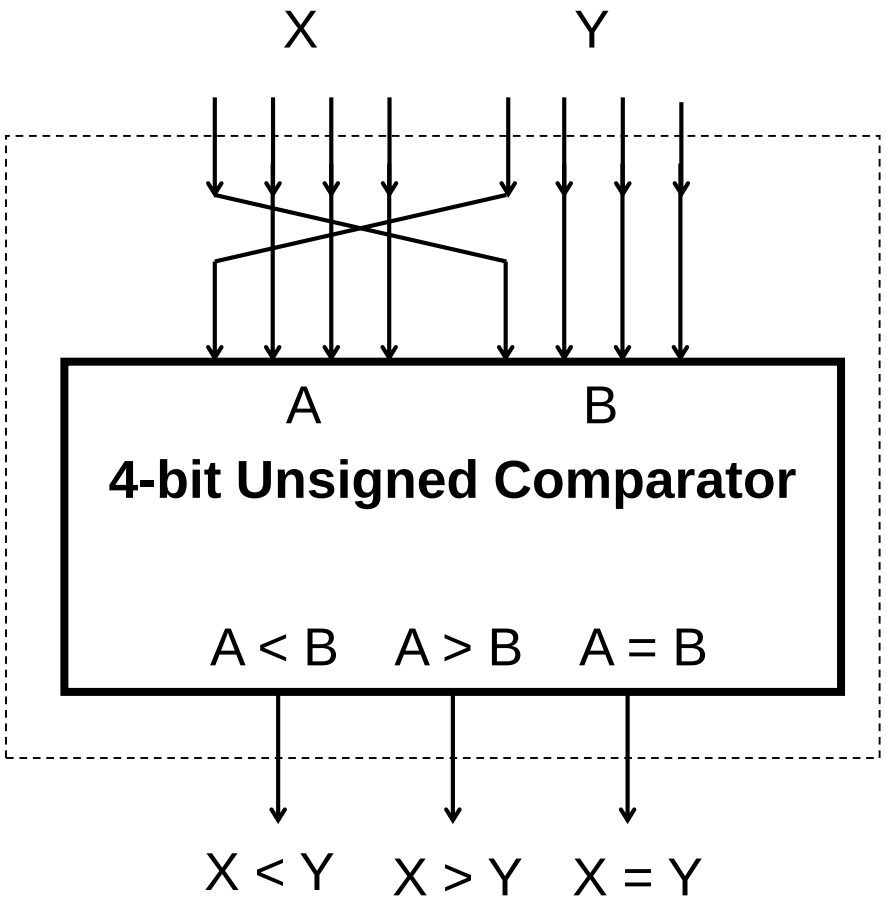
Subtractor as Unsigned Comparator



How to generate Signed A less than B (SALB)?

Convert the unsigned comparator to a signed one

Method I



→ 4-bit Signed Comparator

A B

- 0111 > 0101 (+7 >+ 5)
- 1100 > 1001 (-4 > -7)
- 1111 > 0011 (-1 > +3 NO!)

Swap sign bits

Now unsigned comparator says:

- 0111 < 1011 (-1 < +3 Yes!) or

A < B

which is what a signed comparator should say.

Generate SALB Signal

Method 2 (we learn in this lab)

A – B < 0 (subtraction) means: A < B (comparison)

So we need the sign of the difference, while OV is taken into consideration:

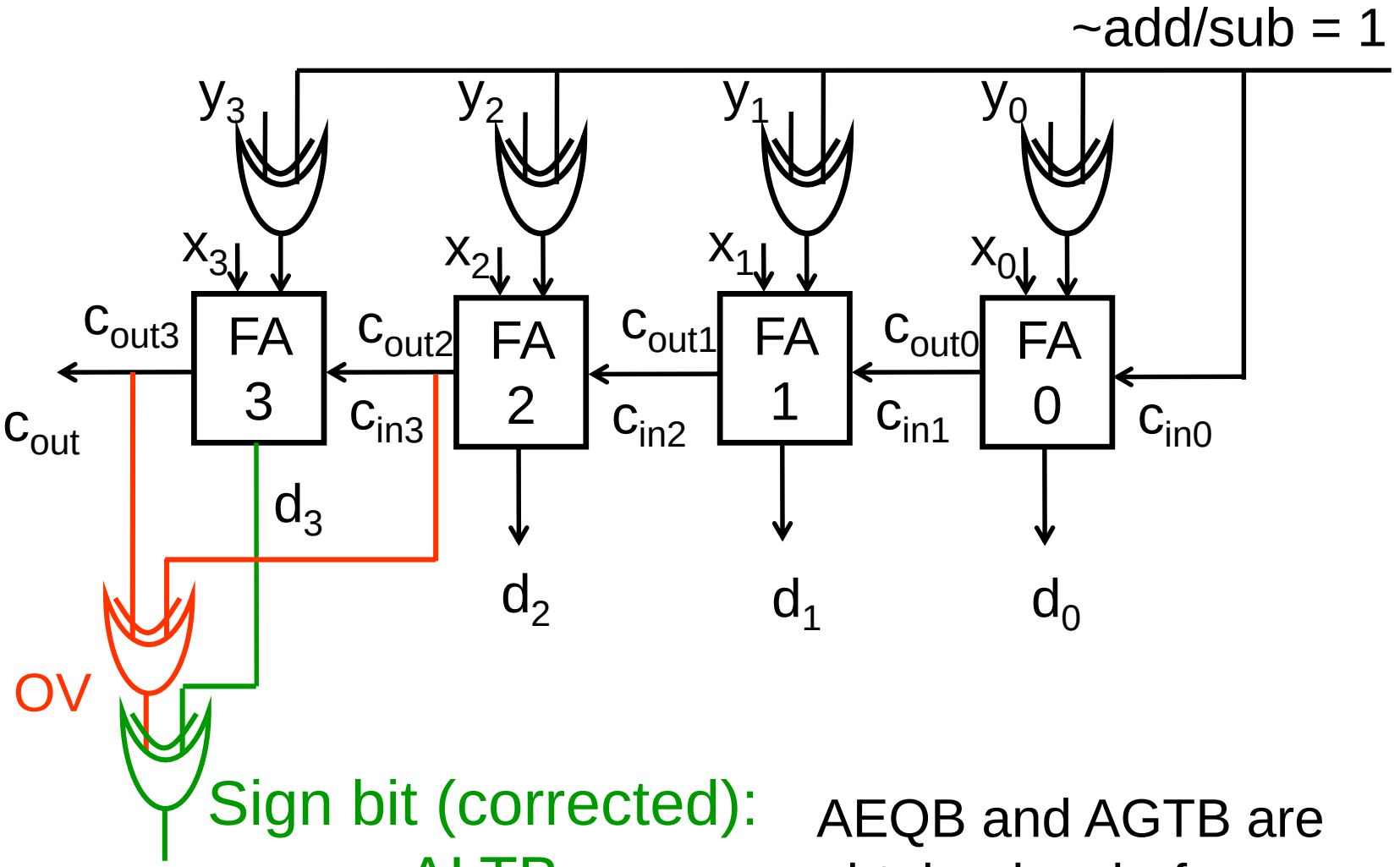
OV	MSb	Sign bit
0	0	0
0	1	1
1	0	1
1	1	0

Sign bit = OV XOR MSb

OV = Cin XOR Cout

Convert the unsigned comparator to a signed one

Method II



AEQB and AGTB are obtained as before.

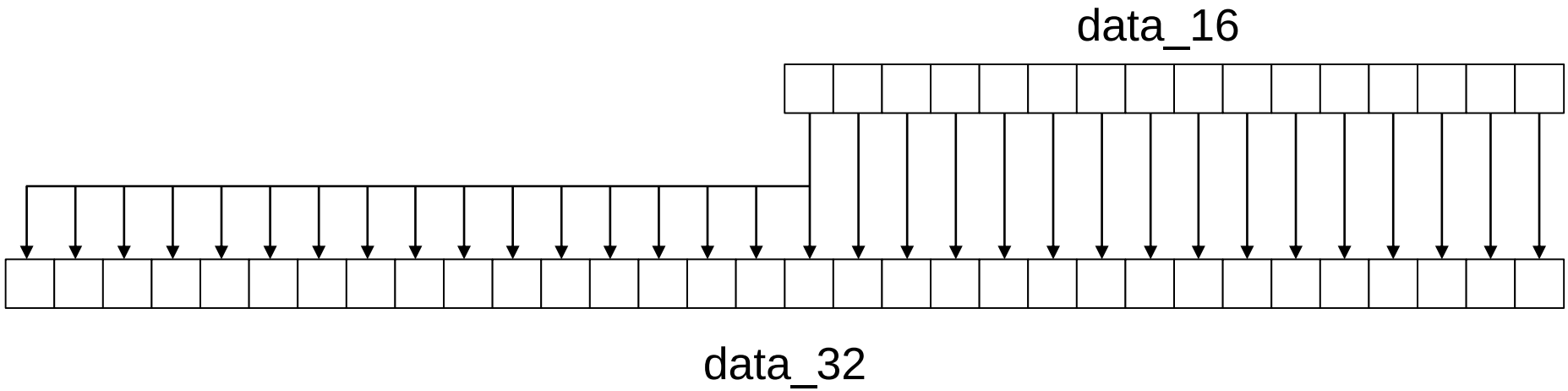
End of Chapter 5, Part II

More features

Concatenation operator:

Shorthand for repeated bits

Example: extend the MSB to get a 32-bit vector from a 16-bit vector




```
data_32 <= (31 DOWNTO 16 => data_16(15)) & data_16;
```

Concatenation operator:

Shorthand for repeated bits

The assignment in the previous slide is equivalent to the following:

```
data_32 <=
data_16(15)&data_16(15)&data_16(15)&data_16(15)&
data_16(15)&data_16(15)&data_16(15)&data_16(15)&
data_16(15)&data_16(15)&data_16(15)&data_16(15)&
data_16(15)&data_16(15)&data_16(15)&data_16(15)&
data_16;
```

Use OTHERS to create a vector with repeated bits:

Example:

The following resets the whole vector to all zero:

```
zero16 <= (OTHERS => '0');
```

Use decimal instead of binary

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```
conv_std_logic_vector(7, 9);
```

converts integer 7 to a std_logic_vector with 9 bits.

Example:

```
Z <= A WHEN m = '1' ELSE conv_std_logic_vector(0, 2);
```