#### CSCI 150 Introduction to Digital and Computer System Design Lecture 3: Combinational Logic Design VI



Jetic Gū 2020 Fall Semester (S3)



#### Overview

- Focus: Arithmetic Functional Blocks
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch4 4.3, 4.7; v5: Ch2 2.9, Ch3 3.10
- Core Ideas:
  - 1. Subtraction I
  - 2. VHDL

#### Review **Unsigned Binary Adder**



#### 1-bit Half Adder

• Half adder input *X*, *Y* output *S*, *C* 







#### 1-bit Half Adder







> • Full adder input *X*, *Y*, *Z*; output S, C



#### 1-bit Full Adder



#### 1-bit Full Adder

• Half adder1 input X, Y output S', C'

• Full adder input *X*, *Y*, *Z*; output S, C

**P0** 

**Binary Adder** 

- Half adder2 input *S'*, *Z* output *S*, *C*"
- C = C' + C''









n-bit Full Adder

**P0 Binary Adder** 





P1 Subtraction

### Unsigned Binary Subtraction I



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows

## Minuend 10110 Subtrahend -100111

Difference



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows0Minuend10110Subtrahend-10011

Difference



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows 10 Minuend 10110 Subtrahend -10011 Difference 1



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows110Minuend10110Subtrahend-10011Difference11



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows0110Minuend10110Subtrahend-10011Difference011



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows00110Minuend10110Subtrahend-10011Difference0011



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows000110Minuend10110Subtrahend-10011Difference00011



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows0000110Minuend101100Subtrahend-100111Difference000111



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

This method works when the Minuend is greater than the Subtrahend!

Borrows	000110	
DOITOWS		Input
Minuend	10110	Dutput
Subtrahend	-10011	
Difference	00011	



#### X > Y, F = X - Y

 We learned to perform subtraction, the greater number

• We learned to perform subtraction, by subtracting the smaller number from



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Borrows0000110Minuend101100Subtrahend-100111Difference000111



### Unsigned 1-bit Binary Subtraction



- Input: Minuend X and Subtrahend YPrevious borrow Z
- Output: Last borrow *B*, difference *D*

Borrows

Minuend X

Subtrahend Y

Difference **D** 

Input Output



#### P1 Subtraction

#### Unsigned 1-bit Binary Subtraction

- Input: Minuend X and Subtrahend YPrevious borrow Z
- Output: Last borrow *B*, difference *D*



X	Y	Ζ	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



#### **Unsigned 1-bit Binary** Subtraction



- Implementation using 3-to-8 Decoder
  - $B = \Sigma m(1,2,3,7)$
  - $D = \Sigma m(1,2,4,7)$





#### Unsigned Binary Subtraction



- Input: Minuend and Subtrahend Previous borrow
- Output: Last borrow, difference

Technology

• 1 bit Unsigned Subtractor

# BorrowsBorrowsConstrained of Constrained of Con







#### Unsigned Binary Subtraction

Technology

• 1 bit Unsigned Subtractor

#### Borrows $B_{000110}$ Minuend $X_{0:n-1}$ 10110 Input Output Subtrahend $Y_{0:n-1}$ -10011Difference $D_{0:n-1}$ 00011







### Hardware Description Language

VHDL (VHSIC-HDL): Very High Speed Integrated Circuit Hardware Description Language





#### What is HDL

- Designing complex circuits using logic circuit diagrams is inefficient
- Hardware Description Language
  - Like programming language, describes hardware structures and behaviours
  - More efficient
  - Common languages
    - Verilog
    - VHDL









File	View Help	
	New	Ctrl+N
	Open	Ctrl+0
	Examples	
	Page Setup	
	Print Setup	
	1 Circuit2.cct	
	2 3-bit Counter.cct	
	3 Simulate.CCT	
_	Exit	Alt+F4



New		ОК
Circuit	~	
Text Document		Cancel
Device Symbol Medel Wissard		



Sim	ulation Model Wizard
	Source
	Create a new, empty model
	This selection will allow you to will generate a shell for the mo
	O Select an existing file Browse
	None Use this setting to create a syn attached. You can use it just f purposes or attach a model late
	Destination
	Open the new model as an indep
	Create a new symbol with the sp
	C Attach the new model to the sele





### VHDL Creating a A

ourc	e
Θ	Create a new, empty model
	This selection will allow you to define the port interface and will generate a shell for the model in the selected format.
0	Select an existing file Browse
0	None
	Use this setting to create a symbol alone, with no model attached. You can use it just for schematic drawing purposes or attach a model later.
)estir	nation
0	Open the new model as an independent design
$\odot$	Create a new symbol with the specified model attached
~	

	<b>ND1INV model</b>		
×			
	Model Info	×	
	Select the desired model type           Structural Circuit         Create a VHDL language file which can be used to describe the function of this device.		
	Enter a name for the new model AND1INV		



- This is where you define all inputs and outputs
  - Input: POS
  - Input: NEG
  - Output: Out1

odel Port Interface		$\times$
Use the controls at right to add pins to the interface list. NOTE: If you are attaching this model to an existing device symbol, the interface list must exactly match the pins on the symbol.	Function  Input  Output  Bidirectional	
	<< Add Single Bit	
	Left Bit Number	
	<< Add Vector	
Drag and drop to re-order items in the list	>> Remove	

3. Type the Name and select the Function accordingly, then press Add Single Bit, click Finish to create the model file.



- This is where you define all inputs and outputs
  - Input: POS
  - Input: NEG
  - Output: Out1

	Use the controls at right to add pins to the interface list. NOTE: If you are attaching this model to an existing device symbol, the interface list must exactly match the pins on the symbol.	Function Input Output Bidirectional
		Name
Name OS	Func Left Right	<< Add Single Bit
IEG )ut1	In Out	Vector Left Bit Number
		Right Bit Number
		<< Add Vector
	Drag and drop to re-order items in the list	>> Remove

3. Type the Name and select the Function accordingly, then press Add Single Bit, click Finish to create the model file.



#### Pin Locations

You can now specify where on the symbol you would like the pins to be placed. To move pins, just drag and drop between the boxes representing the left, top, right and bottom of the symbol.

Left pins NEG POS	Top pins (left to right)	Right pins Out 1
		Symbol Label
	Bottom pins (left to right)	ANDTINV

4. The programme will ask you for Pin Location assignment. Just click Next.



 $\times$ 



```
library IEEE;
  use IEEE.std_logic_ll64.all;
  entity ANDLINV is
   port(
         POS : in std_logic;
         NEG : in std logic;
         Outl : out std logic
     );
  end AND1INV;
  architecture archl of ANDLINV is
  begin
   -- Your VHDL code defining the model goes here
  end arch1;
AND1INV.dwv
```



```
library IEEE;
  use IEEE.std_logic_ll64.all;
 entity ANDIINV is
   port (
         POS : in std_logic;
         NEG : in std logic;
         Outl : out std logic
     );
 end AND1INV;
  architecture archl of ANDLINV is
  begin
    -- Your VHDL code defining the model goes here
  end arch1;
AND1INV.dwv
```



```
library IEEE;
  use IEEE.std_logic_ll64.all;
 entity ANDIINV is
  port (
         POS : in std_logic;
         NEG : in std logic;
         Outl : out std logic
     );
 end AND1INV;
 architecture archl of ANDLINV is
 begin
    -- Your VHDL code defining the model goes here where we implement
  end arch1;
AND1INV.dwv
```



```
library IEEE;
  use IEEE.std logic ll64.all;
  entity AND1INV is
   port (
          POS : in
                    std logic;
                      std logic;
          NEG : in
                  : out std logic
          Outl
     );
  end AND1INV;
  architecture archl of AND1INV is
  begin
    OUT1 <= POS AND NOT NEG AFTER 1NS;
  end archl;
4
Þ
     AND1INV.dwv
```

5. Type: OUT1 <= POS AND NOT NEG AFTER 1NS; This is the Boolean description of the OUT1 port







```
library IEEE;
  use IEEE.std logic ll64.all;
  entity ANDIINV is
   port (
          POS : in
                    std logic;
          NEG : in
                      std logic;
                  : out std logic
          Outl
      );
  end AND1INV;
  architecture archl of AND1INV is
  begin
    OUT1 <= POS AND NOT NEG AFTER 1NS;
  end archl;
4
Þ
     AND1INV.dwv
```

5. Type: OUT1 <= POS AND NOT NEG AFTER 1NS; This is the Boolean description of the OUT1 port









![](_page_40_Picture_3.jpeg)

![](_page_40_Figure_4.jpeg)

![](_page_40_Picture_5.jpeg)

![](_page_41_Figure_1.jpeg)

![](_page_41_Picture_3.jpeg)

![](_page_41_Figure_4.jpeg)

![](_page_41_Picture_5.jpeg)

#### **Run Simulation**

**P2** VHDL

![](_page_42_Picture_2.jpeg)

![](_page_42_Picture_3.jpeg)

![](_page_42_Figure_5.jpeg)

![](_page_42_Picture_7.jpeg)

#### increase value (0 -> 1)

	e	nd a	rch	1;		
	 ¥			DV dame	$\mathbf{h}$	
9	7	Ar		vwb.vv		$\leq$
×	В	rowse.		Script	sVIO	Pane
	Γ	DOC		-		
		neg		z z		
		out1		u		
	•	€ 🕨	$\mathbb{N}$	Timing	λ	VHD

**P2** VHDL

#### 7. Play with the buttons in the I/O panel

![](_page_43_Figure_5.jpeg)

![](_page_43_Picture_6.jpeg)

![](_page_44_Picture_0.jpeg)

### **Run Simulation**

•	end arc	:hl;				
	AND	1INV.dwv				
	Browse	Scripts\IC	)PanelDefai	ult.html		-
					-	
	pos	z	+	-	0	
	neg	z	+	-	0	
	out1	u	+	-	0	
	,					
		\ Timing  \		O Panel /		

8. Changes are reflected in the Timing Diagram. Use Zoom panel to Zoom In and Out

![](_page_44_Figure_4.jpeg)

![](_page_44_Picture_5.jpeg)

![](_page_44_Figure_6.jpeg)

![](_page_44_Picture_8.jpeg)

#### Exe1: 1-bit Half Adder

- Create a new component in VHDL called HalfAdder1
  - Input: X, Y

**P2** 

VHDL

- Output: S, C
- **Don't use** AFTER

![](_page_45_Figure_6.jpeg)

![](_page_45_Picture_7.jpeg)

![](_page_45_Picture_8.jpeg)

![](_page_45_Picture_9.jpeg)

![](_page_45_Picture_10.jpeg)

![](_page_46_Picture_0.jpeg)

#### architecture arch1 of HalfAdder is

#### begin

 $S \ll X X OR Y;$ 

 $C \ll X AND Y;$ 

end arch1;

#### Exe1: 1-bit Half Adder

![](_page_46_Figure_8.jpeg)

![](_page_46_Picture_9.jpeg)

![](_page_46_Picture_10.jpeg)

![](_page_46_Picture_11.jpeg)

![](_page_46_Picture_12.jpeg)