

CSE140 L

Instructor: *Thomas Y. P. Lee*
January 18, 2006



CSE140L Course Info

- **Lectures**
 - Wednesday 10:00-11:20AM, HSS1330
- **Lab 1 Assignment Begins**
 - ◆ TA's JinHua Liu (jhliu@cs.ucsd.edu)
 - ◆ Contact TAs if you're still looking for a lab partner.
 - ◆ Lab group list is maintained on the webboard.
 - ◆ TA office hour will be posted on the class webpage.
- **Office Hours**
 - ◆ E-mail: yunparn_lee@yahoo.com (may change later)
 - ◆ Office Hours: appointment only
- **Lab Hours Start from next week**
 - ◆ When: check website for detail schedule
 - ◆ Where: B230, B240, B250, B260, B270
 - ◆ You may work on your own PC. VHDL or Verilog both accepted.
- **Course Policy**
 - ◆ Honesty is strictly enforced. You may only discuss project with your lab partner.



Agenda



- **Lab 1**
 - ◆ *Number Systems*
 - ◆ *2's Complement Arithmetic*
 - ◆ *Overflow Condition*
 - ◆ *Binary Half/Full Adder*
 - ◆ *Ripple Carry Adder*
 - ◆ *2's Complement Adder/Subtractor*
 - ◆ *Comparator*
 - ◆ *Multiplexer, Logic Diagram, 4 Inputs Multiplexer*
 - ◆ *2-to-4 Demultiplexer*
- **Design Tools - QuartusII**
 - ◆ *HDL, Design Entry and Design Process*
 - ◆ *Synthesis and FPGA's Macro*
 - ◆ *Place and Route in FPGA*
 - ◆ *Time Delay in Digital Circuit*
 - ◆ *Timing Analysis – Worst (critical) Case Delay*
 - ◆ *QuartusII Design Flow*
 - ◆ *Floorplanner*
 - ◆ *QuartusII Tutorial*

Number System



- Three common schemes: sign-magnitude, one's complement, two's complement
- Sign-magnitude: MSB=0 for positive, 1 for negative
 - Range: $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$
 - Two representations for zero: 0000..&1000..
 - Simple multiplication but complicated addition/subtraction
- One's complement: If N is positive, then the negative is /N
 - Example: 0111 = 7, 1000 = -7
 - Range: $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$
 - Two representation for zero: 0000..& 1111..
 - Subtraction implemented as addition followed by one's complement

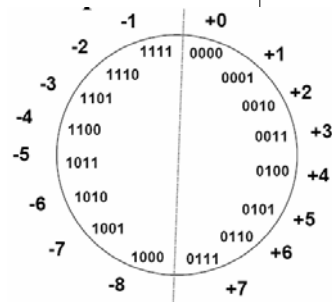
Two's Complement Representation

- Two's complement = bitwise complement + 1

$$0111 \rightarrow 1000 + 1 = 1001 = -7$$

$$1001 \rightarrow 0110 + 1 = 0111 = 7$$

- Asymmetric range: -2^{N-1} to $+2^{N-1}-1$
- Only one representation for zero
- Simple addition and subtraction
- Many advantages, most common representation

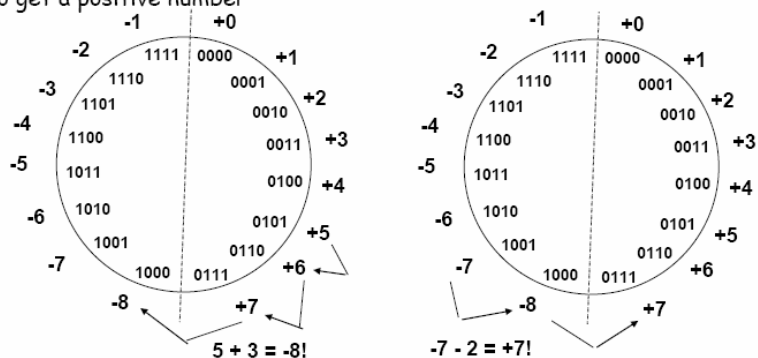


4	0100	-4	1100	4	0100	-4	1100
+ 3	0011	+ (-3)	1101	- 3	1101	+ 3	0011
7	0111	-7	11001	1	10001	-1	1111

[Katz93, chapter 5]

What is Overflow Condition in Addition/Subtraction?

Add two positive numbers to get a negative number or two negative numbers to get a positive number



$$\begin{array}{r} 5 \quad 0101 \\ + 3 \quad 0011 \\ \hline -8 \quad 1000 \end{array}$$

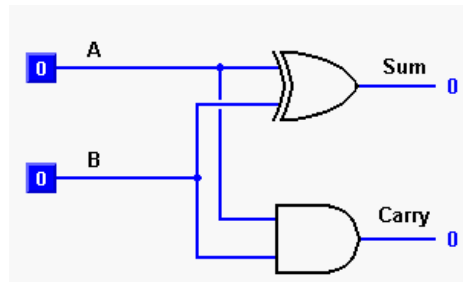
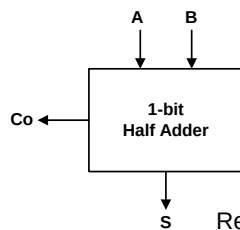
$$\begin{array}{r} -7 \quad 1001 \\ - -2 \quad 1100 \\ \hline 7 \quad 0111 \end{array}$$

If carry in to sign equals carry out, then can ignore carry out, otherwise have overflow

1 Bit- Half Adder

- 1-Bit Half Adder -Addend, augend as inputs, S and Co as outputs
 - Truth Table is showing below

Inputs		Outputs	
A	B	S	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Reference: <http://www.play-hookey.com/digital/adder.html>

Full Adder

- Full Adder –Addend, augend, Ci as inputs, S and Co as outputs
 - Truth Table is showing below

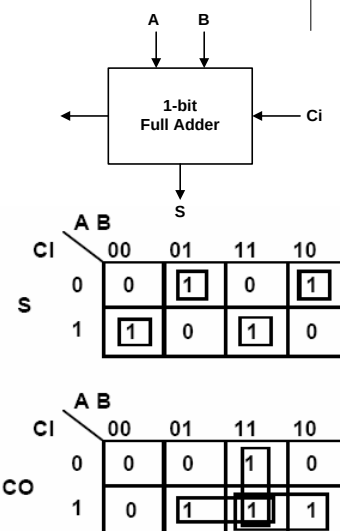
Inputs			Outputs	
A	B	Ci	S	Co
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Sum of Product Canonical Form

$$S = A \oplus B \oplus C_i$$

$$= \overline{A}\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\overline{C}_i + ABC_i$$

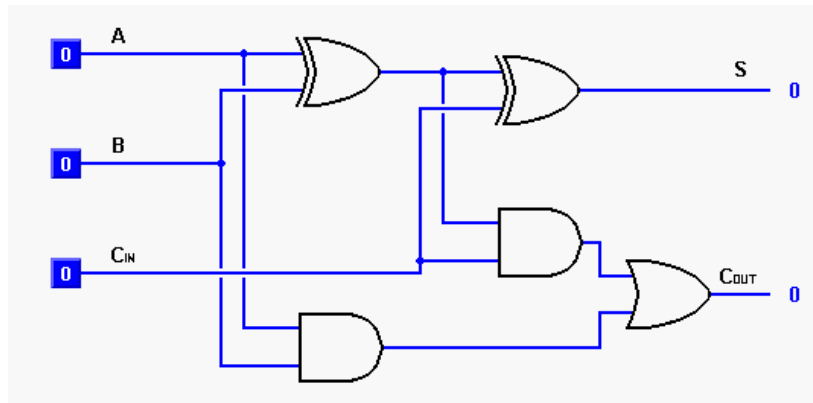
$$C_o = AB + C_i(A+B)$$



Block Diagram of Full Adder



- 1-Bit Full Adder –Addend, augend, C_i as inputs, S and C_o as outputs
 - Truth Table is showing below



Reference: <http://www.play-hookey.com/digital/adder.html>

Verilog Example of Full Adder

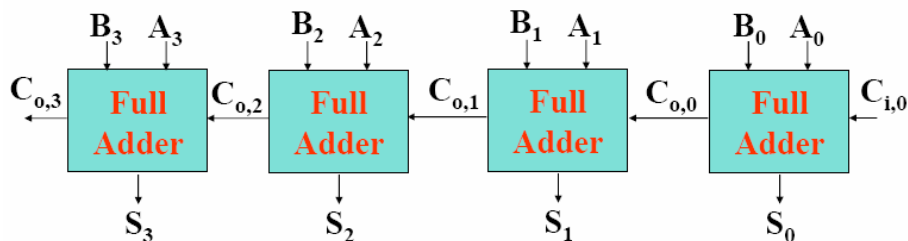


- Module define the input, output signals
- Assign statement contains logic operation of a full adder

```
module full_adder ( a, b, cin, sum, cout);  
    input          a, b, cin;  
    output         sum, cout;  
  
    assign sum = a ^ b ^ cin;  
    assign cout = ( a & b ) | ( cin & ( a | b ) );  
endmodule
```

Note: Operators (&, |, ...) will be explained in the next session
Also see Verilog References and VHDL references

Ripple Carry Adder

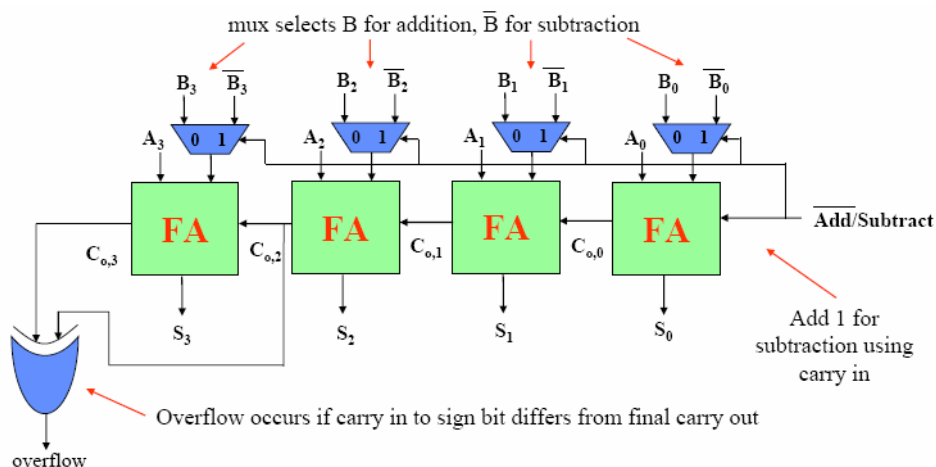


- Worst case propagation delay is related to number of bits

$$t_{\text{adder}} = (N-1) \times t_{\text{carry}} + t_{\text{sum}}$$

2's Complement Subtractor

Subtracting a number is the same as adding the bitwise complement of number then adding 1, combination of addition & subtraction is shown below:



Comparator



2-Bit Comparator

- Compares two 2-bit unsigned number, output the relationship

LE = 1 if $A < B$, LE = 0 otherwise

EQ = 1 if $A = B$, EQ = 0 otherwise

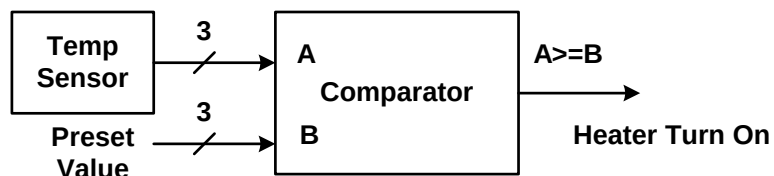


A	B	LE	EQ
00	00	0	1
01	01	0	1
01	10	1	0
10	01	0	0
11	11	0	1

Comparator- Example of Combinational Circuits



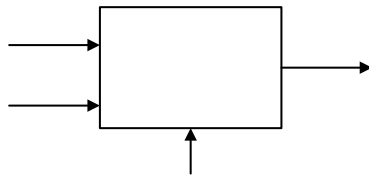
- Output is a function of current input
- Example - Thermostat



Multiplexer



- Multiplexing is converting several inputs to single output according to control (select) lines
- 2-to-1 Multiplexer
 - Select one of the input signals according to the control signals, truth table is shown below

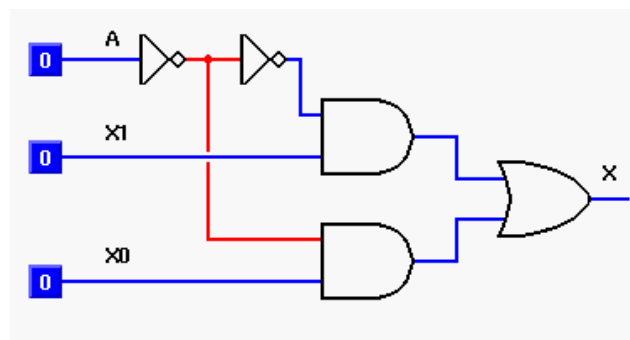


A	B	Sel	Out
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

Multiplexer Logic Diagram



- Multiplexing is converting several inputs to single output according to control (select) lines
- 2-to-1 Multiplexer
 - Select one of the input signals according to the control signals, truth table is shown below

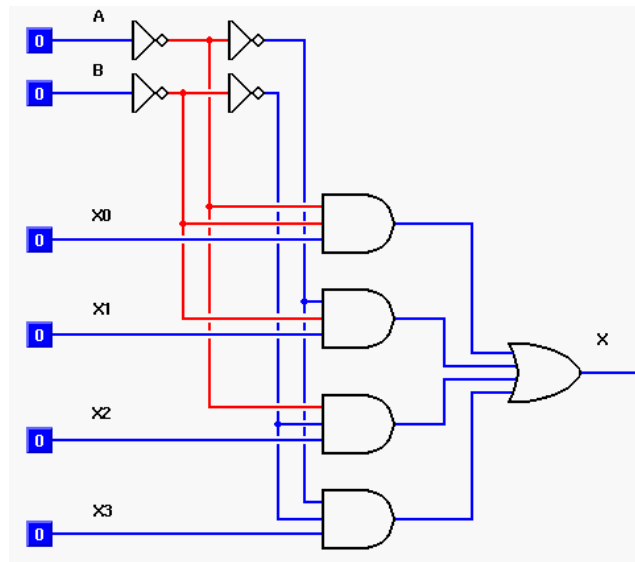


Reference: <http://www.play-hookey.com/digital/multiplexer.html>

Out

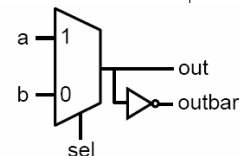
4 Inputs Multiplexer Logic Diagram

- 4-to-1 Multiplexer: 2 Select control lines



An Example of Verilog of 2-to-1 Multiplexer

- Verilog designs consist of interconnected **modules**
- A module can be an element or lower level design block
- ?: means conditional operator



```
module mux_2_to_1(a, b, out,
                  outbar, sel);
```

Declare and name a module; list its ports. Don't forget that semicolon.

```
// This is 2:1 multiplexor
```

Comment starts with //
Verilog skips from // to end of the line

```
input a, b, sel;
output out, outbar;
```

Specify each port as input, output, or inout

```
assign out = sel ? a : b;
assign outbar = ~out;
```

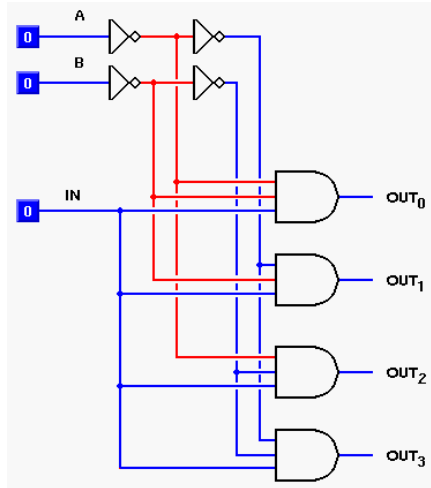
Express the module's behavior. Each statement executes in parallel; order does not matter.

```
endmodule
```

Conclude the module code.

2-to-4 Demultiplexer (Decoder)

- n-Bit binary number decode to 2^n output lines, 3-to-8 decoder, 4-to-16 decoder, useful in memory decoding



http://www.play-hookey.com/digital/decoder_demux_four.html

HDL and Design Process

- Hardware Description Language (HDL) is a convenient, device-independent representation of digital logic
- The most common HDL – Verilog and VHDL
- HDL description is compiled into **NETLIST**
- Synthesis optimizes the logic
- Mapping the target **NETLIST** to a specific hardware platform

Verilog

```
input a,b;
output sum;
assign sum <= {1b'0, a} + {1b'0, b};
```

Compilation and Synthesis

Netlist

```
g1 "and" n1 n2 n5
g2 "and" n3 n4 n6
g3 "or" n5 n6 n7
```

Mapping

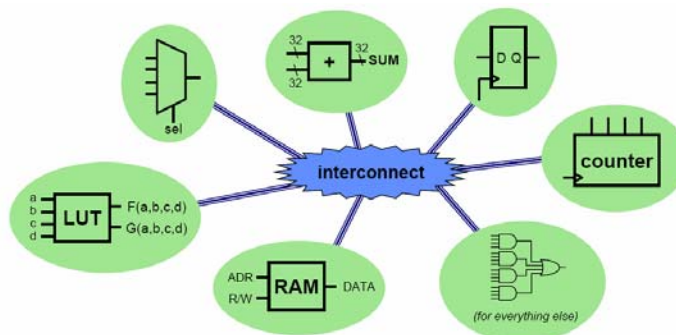
FPGA

PAL

ASIC
(Custom ICs)

Synthesis and FPGA's Macros

- FPGA Build-in components are called **"Macro"**
- FPGA is like an electronic breadboard that is wired together by an **Automated Synthesis Tool**



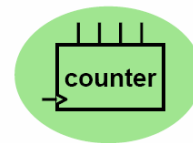
Place and Route in FPGA

- **Optimal choose the FPGA macros that can efficiently implement various parts of the HDL code**

```
...
always @ (posedge clk)
begin
    count <= count + 1;
end
...
```

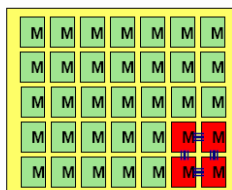
HDL Code

"This section of code looks like a counter. My FPGA has some of those..."



Inferred Macro

- **Place-and Route: consider area and/or speed factor, choose the needed macros by location and route the interconnect**



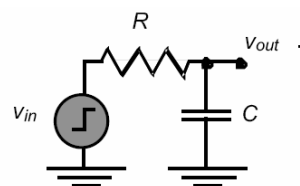
"This design only uses 10% of the FPGA. Let's use the macros in one corner to minimize the distance between blocks."

Floorplaner



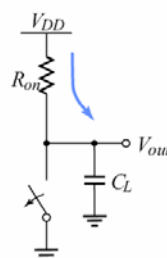
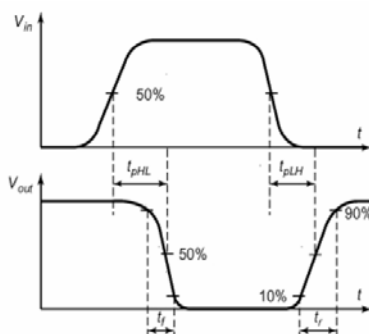
- Floorplanner is an interactive graphical tool that allows edit/view location constraints in design.
- Floorplanning is an optional methodology to improve automatic place and route. Particularly useful on structured designs and data path logic.

Time Delay in Digital Circuits

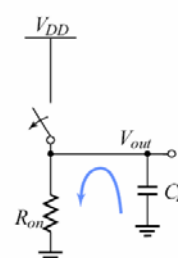


$$v_{out}(t) = (1 - e^{-t/\tau}) V$$

$$t_p = \ln(2) \tau = 0.69 RC$$

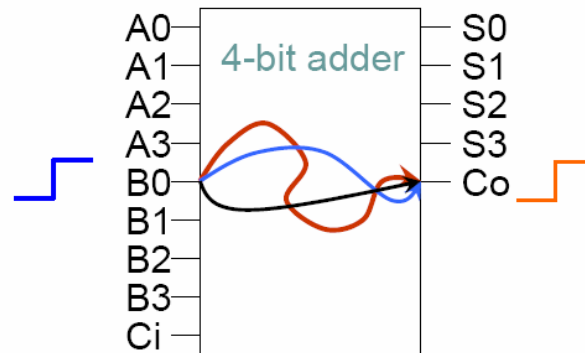


(a) Low-to-high



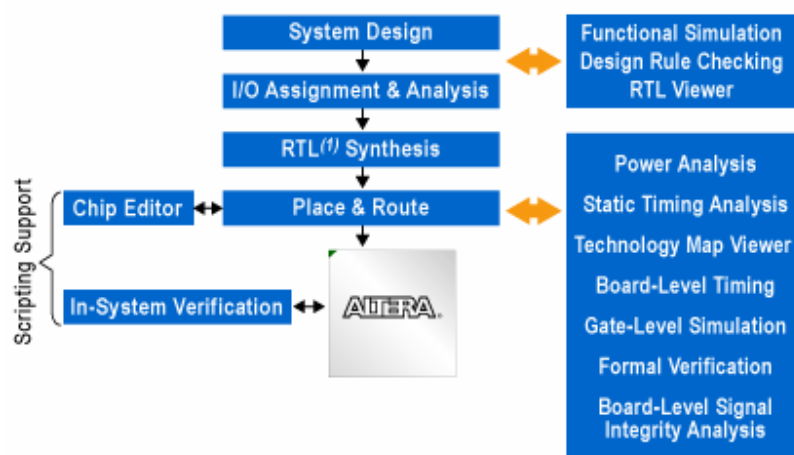
(b) High-to-low

Worst Case Time Delay in Adder



worst case delay: B0 to Co

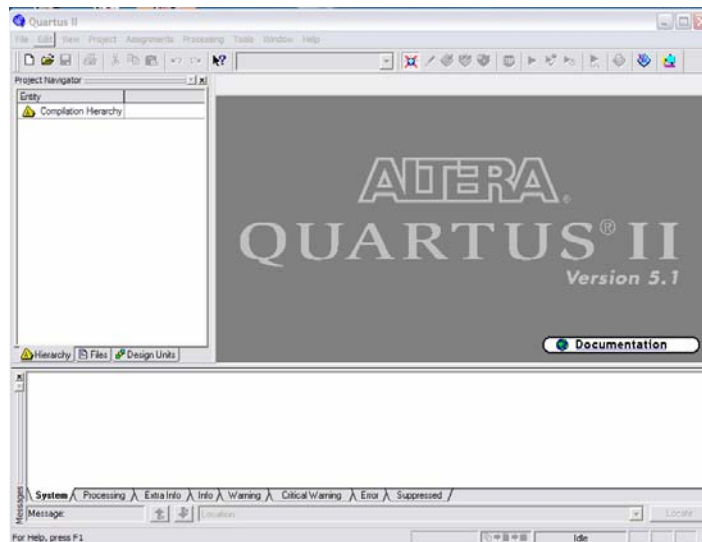
QuartusII Design Flow



<http://www.altera.com/products/software/flows/fpga/flo-fpga.html>

QuartusII Tutorial

- Create New Design Entry – Verilog, VHDL or Schmatic Diagram



Create Project of Adder/Subtractor

- Creating and Compiling Projects
- Setting Design Constraints using the assignment editor
- Analyzing clock and I/O timing
- Adding timing constraints to achieve better design performance
- Simulating design using QuartusII Simulator
- Configuring Altera FPGA device

