Weeks 3 & 4 Outline

- Multiple output SOP designs
- Verilog implementations
  - using SOP
  - using conditional statements
  - using case statements
- Logic operations on number systems
- More examples of combinational logic
- Exam 1 – Tuesday, Sept. 25th

Addition Tables

Decimal

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Binary

+	0	1
0	0	1
1	1	0

Denotes Carry Operation

Addition Table for Hexadecimal

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
3	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
4	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3
5	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4
6	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5
7	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8
A	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
B	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A
C	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B
D	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C
E	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D
F	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E

- denotes carry out is set

One Bit Addition (Half Adder)

- Arithmetically:

  Z = A + B

- Logically (Verilog):

  Z = A ^ B; (xor)

  C = A & B;

Inputs		**Outputs**	
A	**B**	**Z**	**C**
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Multi-bit (Binary) Addition

- Compare to decimal addition

```
  14
+  7
  21
```

- Binary addition (expand to 5 bits)

```
  11    (carries)
 01110
+00111
 10101
```


Multi-bit (Binary) Addition (cont.)

- Addition examples:

Z = A + B;

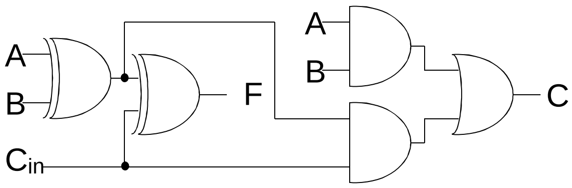
Carry	0	1	1	0	1	1	1	x	Decimal
A	0	0	0	1	0	1	1	1	23
B	0	0	1	1	0	0	0	1	49
Z	0	1	0	0	1	0	0	0	72

Carry	1	1	1	1	0	1	1	x	Decimal
A	0	1	0	0	1	0	1	1	75
B	0	0	1	1	1	0	0	1	57
Z	1	0	0	0	0	1	0	0	132

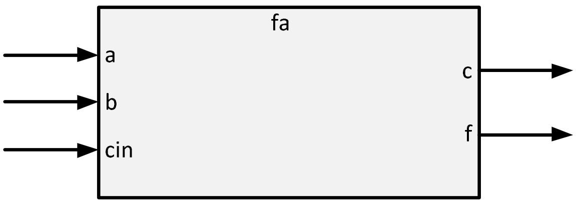
One Bit Addition (Full Adder)

- Addition: F = A + B (with carry)

Inputs			Outputs	
A	B	C_{in}	F	C
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1



Full Adder – Block Diagram & Verilog Module



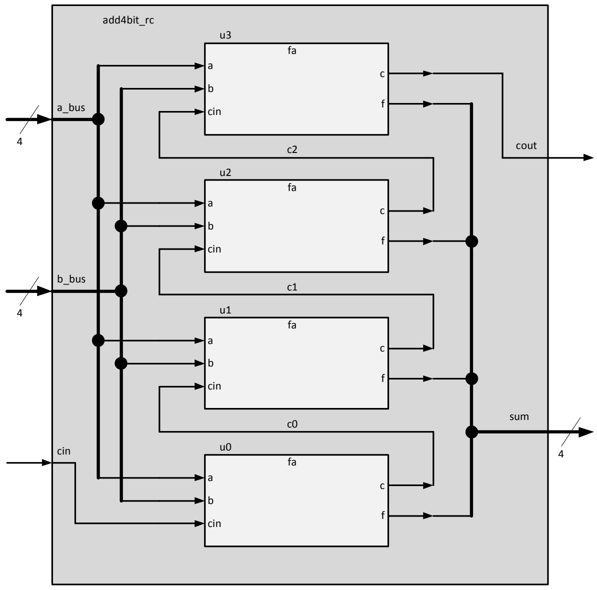

```
module fa (a,b,cin,f,c);
  input a,b,cin;
  output f,c;
…

…
endmodule
```

Full Adder – Block Diagram & Verilog Module

- Implementation in class or as homework

Four Bit Adder Block Diagram (Ripple Carry)



Four Bit Adder Block Diagram (Ripple Carry)

- Implementation in class or as homework

```
module add4bit_rc (a_bus, b_bus, cin, sum, cout);
       input [3:0]a_bus, b_bus;
       input cin;
       output [3:0]sum;
       output cout;

       // define the nets needed for interconnection

       // instantiate the full adders
…

…
endmodule
```


Signed Integers

- Binary numbers can represent both positive and negative values.
- For an *n-bit* number, there are still 2^n possible values, but now they are divided as half positive (including 0) and half negative.

Signed Integer Representations

- Signed magnitude: the most significant bit is used as the sign bit, e.g.:

  $3_{10} = 0011_2$ $-3_{10} = 1011_2$
- 1's complement: change 0's to 1's and 1's to 0's (C code symbol: ~)

  $3_{10} = 0011_2$ $-3_{10} = 1100_2$
- Many problems with both:
  – have two representations for 0
  – difficult to do simple binary arithmetic

Signed Integer Representations

- All processors use what is called 2's complement representation
- The conversion rule:
  – take the 1's complement of the binary number
  – add 1 to the binary number

$3_{10} = 0011_2$ $\sim 0011_2 = 1100_2$ $-3_{10} = 1101_2$

2's Complement Notation

- Example, 4 bit number:
  - 16 values
  - Unsigned: 0 to 15
  - Signed: -8 to +7
- One minor problem:
  - asymmetric in max & min values:
    - max: $2^{n-1} - 1$
    - min: -2^{n-1}

Value	Unsigned Integer	Signed Integer
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

2's Complement Notation

- A major advantage: arithmetic works seamlessly
  - addition
  - subtraction
- Examples:

2	0010	6	0110	-6	1010	5	0101
-2	1110	-2	1110	-2	1110	+2	0010
0	0000	4	0100	-8	1000	7	0111

2’s Complement Sign Extension

- To extend a signed number from *n* bits to *m* bits:
  - add *m* – *n* bits to the msb side
  - fill the extra bits with the value of the original msb
- Examples 4 bits to 8 bits:

Number	4 bit				8 bit							
2	0	0	1	0	0	0	0	0	0	0	1	0
-2	1	1	1	0	1	1	1	1	1	1	1	0

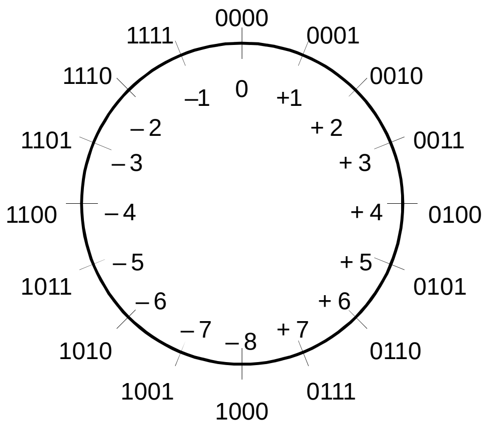
2’s Complement in Hexadecimal

- Same rules, but different symbols
  - find the complement (~) of each symbol
  - add 1 to the hexadecimal number

Hex Symbol	~(Hex Symbol)	~(Hex Symbol) + 1
0	F	0
1	E	F
2	D	E
3	C	D
4	B	C
5	A	B
6	9	A
7	8	9
8	7	8
9	6	7
A	5	6
B	4	5
C	3	4
D	2	3
E	1	2
F	0	1

Circular Representation of 2's complement, 4-bit numbers.

- 4-bit range, 16 numbers: -8 to +7
- Clockwise for addition; CCW for subtraction



Parity

- Easiest implemented with XOR gates (remember our adders)
  - Example: 4 bit data to generate parity bit:

    P = a[3] ^ a[2] ^ a[1] ^ a[0]; // for even

    P = ~(a[3] ^ a[2] ^ a[1] ^ a[0]); // for odd
  - Data checking – 5 bit data checking bit:

  check_bit = P ^ a[3] ^ a[2] ^ a[1] ^ a[0];
- If even parity: check_bit should be 0
- If odd parity: check_bit should be 1

Gate Completeness

- For any Boolean function need: AND, OR, NOT
- What can be done with a NAND gate?
  - NOT operation:
    - Connect both inputs together
  - AND operation
    - Connect output of NAND to input of NOT
  - OR operation
    - Connect NOT's to inputs of NAND

Binary Addition (cont.)

- Addition – expressions

$A/B/C_{in}$ result (carry)

0 + 0 + 0 = 0 (0)

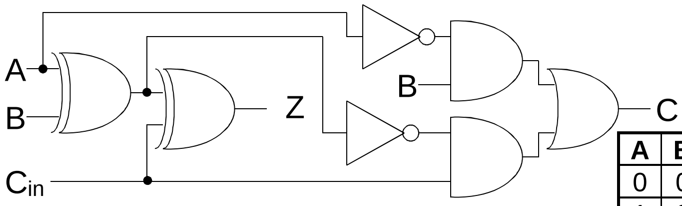
1 + 0 + 0 = 1 (0)

1 + 1 + 0 = 0 (1)

1 + 1 + 1 = 1 (1)

Binary Arithmetic (cont.)

- Subtraction (carry label is now a borrow)



A	B	C_{in}	Z	C
0	0	0	0	0
1	0	0	1	0
0	1	0	1	1
1	1	0	0	0
0	0	1	1	1
1	0	1	0	0
0	1	1	0	1
1	1	1	1	1

- Z = A – B (with borrow)

Binary Arithmetic

- Subtraction examples:

Z = A - B;

Carry	1	1	0	0	0	0	0	x	Decimal
A	0	0	0	1	0	1	1	1	23
B	0	0	1	1	0	0	0	1	- 49
Z	1	1	1	0	0	1	1	0	-26

Carry	0	1	1	0	0	0	0	x	Decimal
A	0	1	0	0	1	0	1	1	75
B	0	0	1	1	1	0	0	1	- 57
Z	0	0	0	1	0	0	1	0	18

Hexadecimal Arithmetic

- Use:
  a) tables or
  b) converting to decimal, and operating one digit at a time.
- Example of b):
  Z = 0xBC4A + 0x3879
  ...
  Z = 0xF4C3

Hex table

Carry	1	0	1	x
A	B	C	4	A
B	3	8	7	9
C	F	4	C	3

Decimal conversion

Carry	1	0	1	x
A	11	12	4	10
B	3	8	7	9
C	15	20	12	19
C mod 16	15	4	12	3

End of combinational logic lectures

- Review … Exam 1