Computer Architecture
(计算机体系结构)

Lecture 23 – Combinational Logic Blocks

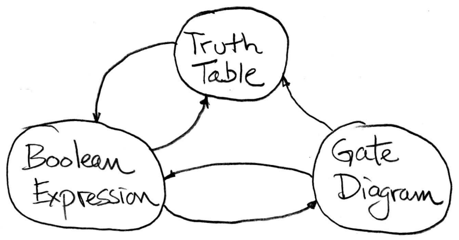
2020-10-12

Lecturer Yuanqing Cheng

www.cadetlab.cn/~course

Review

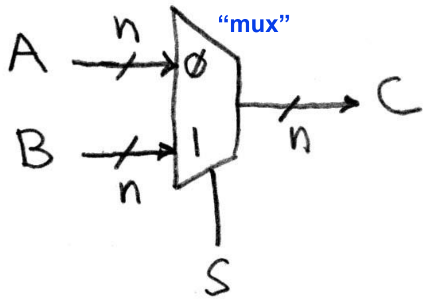
- Use this table and techniques we learned to transform from 1 to another



Today

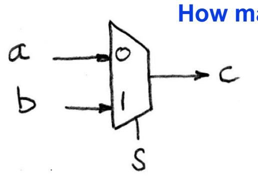
- Data Multiplexors
- Arithmetic and Logic Unit
- Adder/Subtractor

Data Multiplexor (here 2-to-1, n-bit-wide)



N instances of 1-bit-wide mux

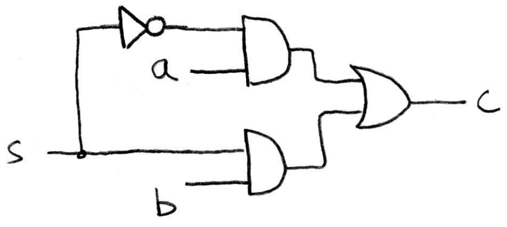
How many rows in TT?



$$\begin{aligned} c &= \bar{s}a\bar{b} + \bar{s}ab + s\bar{a}b + sab \\ &= \bar{s}(a\bar{b} + ab) + s(\bar{a}b + ab) \\ &= \bar{s}(a(\bar{b} + b)) + s((\bar{a} + a)b) \\ &= \bar{s}(a(1) + s((1)b) \\ &= \bar{s}a + sb \end{aligned}$$

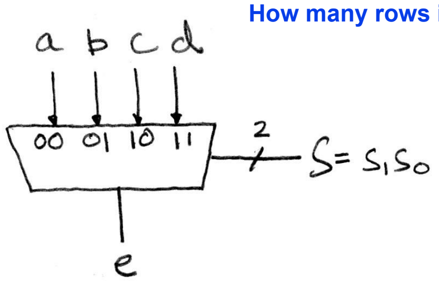
How do we build a 1-bit-wide mux?

$$\bar{s}a + sb$$



4-to-1 Multiplexor?

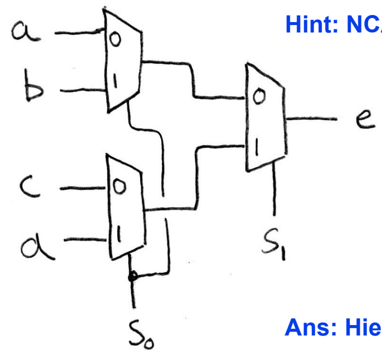
How many rows in TT?



$$e = \overline{s_1}\,\overline{s_0}a + \overline{s_1}s_0 b + s_1\overline{s_0}c + s_1 s_0 d$$

Is there any other way to do it?

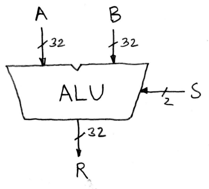
Hint: NCAA tourney!



Ans: Hierarchically!

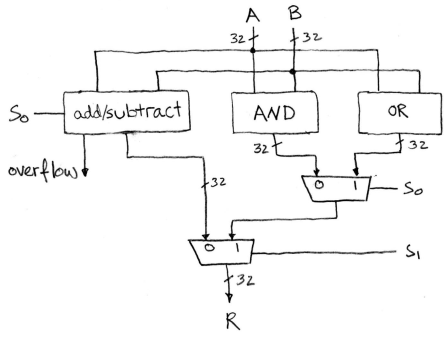
Arithmetic and Logic Unit

- Most processors contain a special logic block called “Arithmetic and Logic Unit” (ALU)
- We’ll show you an easy one that does ADD, SUB, bitwise AND, bitwise OR



when S=00, R=A+B
when S=01, R=A-B
when S=10, R=A AND B
when S=11, R=A OR B

Our simple ALU



Adder/Subtracter Design -- how?

- Truth-table, then determine canonical form, then minimize and implement as we’ve seen before
- Look at breaking the problem down into smaller pieces that we can cascade or hierarchically layer

Adder/Subtracter – One-bit adder LSB…

$$\begin{array}{rccc|c|}
 & a_3 & a_2 & a_1 & a_0 \\
+ & b_3 & b_2 & b_1 & b_0 \\
\hline
 & s_3 & s_2 & s_1 & s_0
\end{array}$$

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$s_0 =$
$c_1 =$

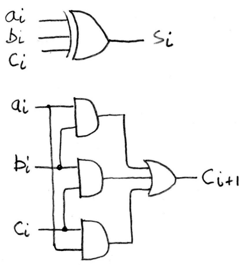
Adder/Subtracter – One-bit adder (1/2)…

$$\begin{array}{cccccc} & a_3 & a_2 & \boxed{a_1} & a_0 \\ + & b_3 & b_2 & \boxed{b_1} & b_0 \\ \hline & s_3 & s_2 & \boxed{s_1} & s_0 \end{array}$$

a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

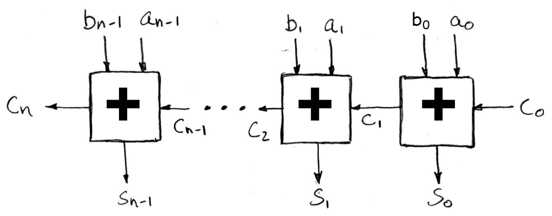
$$s_i =$$
$$c_{i+1} =$$

Adder/Subtracter – One-bit adder (2/2)…



$$s_i = \text{XOR}(a_i, b_i, c_i)$$
$$c_{i+1} = \text{MAJ}(a_i, b_i, c_i) = a_ib_i + a_ic_i + b_ic_i$$

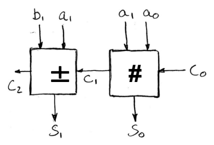
N 1-bit adders ⇒ 1 N-bit adder



What about overflow?
Overflow = c_n?

What about overflow?

- Consider a 2-bit signed # & overflow:
  - `10 = -2 + -2 or -1`
  - `11 = -1 + -2 only`
  - `00 =  0 NOTHING!`
  - `01 =  1 + 1 only`



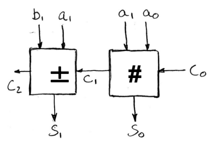
- Highest adder
  - C_1 = Carry-in = C_{in}, C_2 = Carry-out = C_{out}
  - No C_{out} or C_{in} ⇒ NO overflow!
  - C_{in}, and C_{out} ⇒ NO overflow!
  - C_{in}, but no C_{out} ⇒ A,B both > 0, overflow!
  - C_{out}, but no C_{in} ⇒ A,B both < 0, overflow!

What op?

What about overflow?

- Consider a 2-bit signed # & overflow:

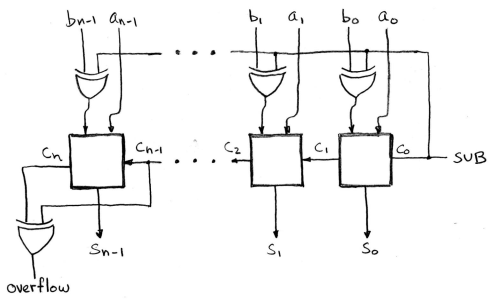
```
10 = -2
11 = -1
00 =  0
01 =  1
```



- Overflows when…
  - C_{in}, but no C_{out} ⇒ A,B both > 0, overflow!
  - C_{out}, but no C_{in} ⇒ A,B both < 0, overflow!

$$\text{overflow} = c_n \text{ XOR } c_{n-1}$$

Extremely Clever Subtractor



Peer Instruction

1) Truth table for mux with 4-bits of signals has 2^4 rows

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl

	12
a)	FF
b)	FT
c)	TF
d)	TT

Peer Instruction Answer

1) Truth table for mux with 4-bits of signals controls 16 inputs, for a total of 20 inputs, so truth table is 2^{20} rows… FALSE

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl … TRUE

1) Truth table for mux with 4-bits of signals is 2^4 rows long

2) We could cascade N 1-bit shifters to make 1 N-bit shifter for sll, srl

	12
a)	FF
b)	FT
c)	TF
d)	TT

"And In conclusion…"

- Use muxes to select among input
  - S input bits selects 2^S inputs
  - Each input can be n-bits wide, indep of S
- Can implement muxes hierarchically
- ALU can be implemented using a mux
  - Coupled with basic block elements
- N-bit adder-subtractor done using N 1-bit adders with XOR gates on input
  - XOR serves as conditional inverter