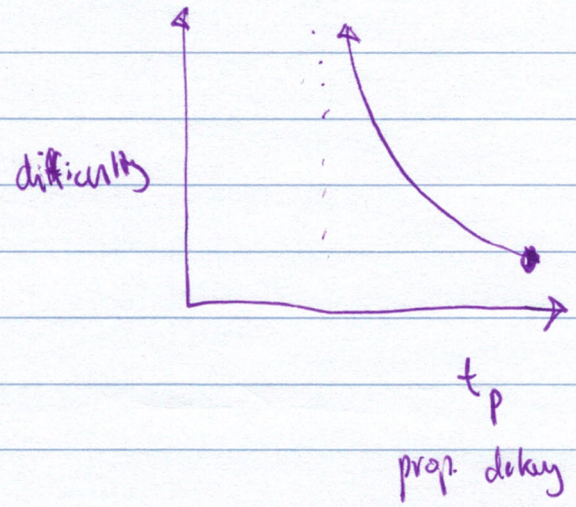
EEC 18

Dec. 2

What if requirement is violated?

A) Design time

- speed up logic



B) After chip is built

- $t_{clk\text{-}to\text{-}q}$, t_{logic}, t_{setup} can not change

- can change f_{clk}

  i) Commercial product - maybe ok

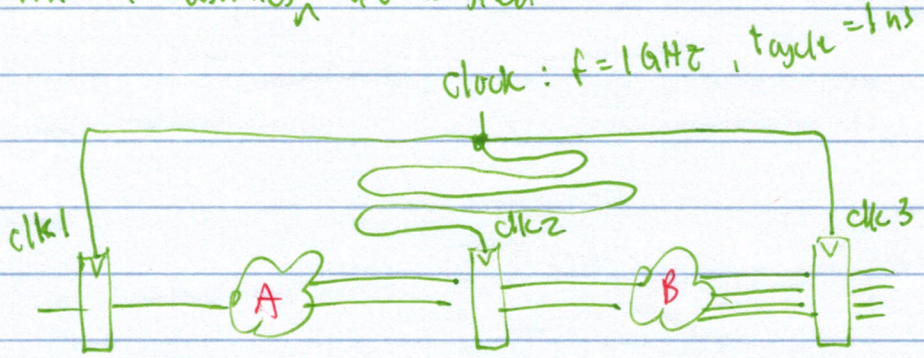
  - re-burned at lower clock

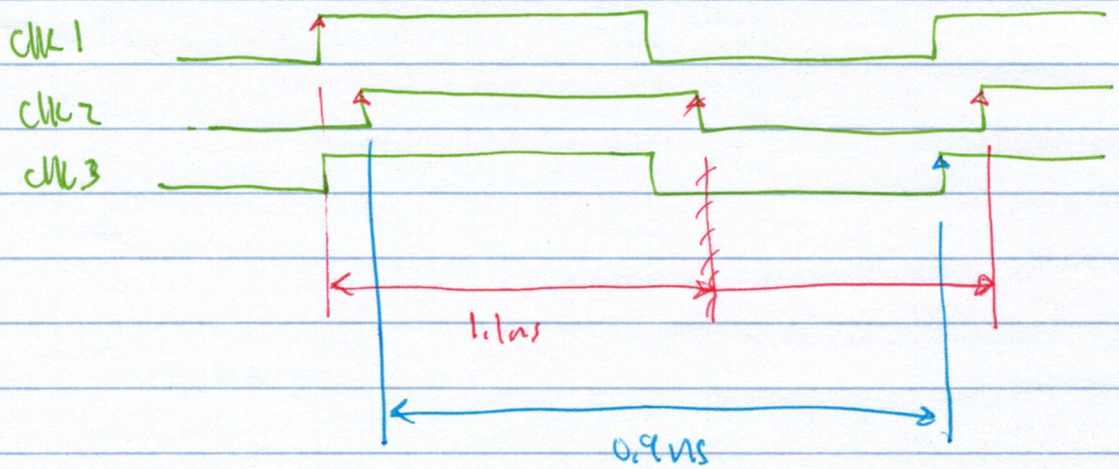
  - real-time e.g. 60 f/s, 59.9 f/s not ok

  ii) research

  - probably ok

Equation #1 assumes clks are aligned

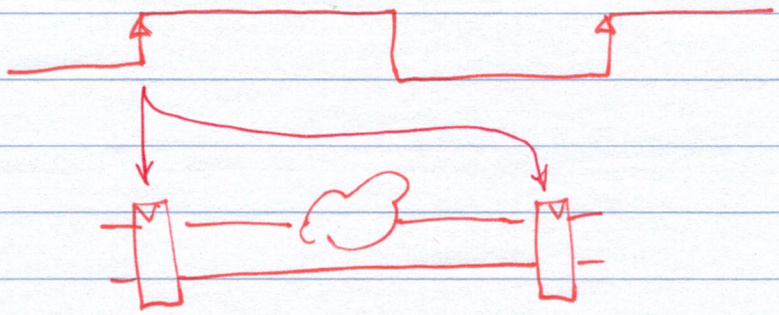




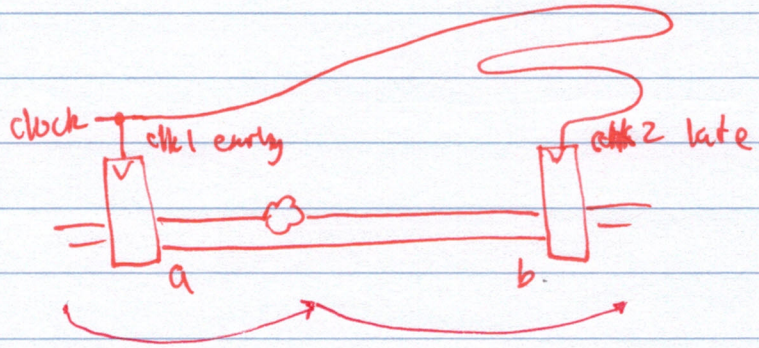
For simple analysis,

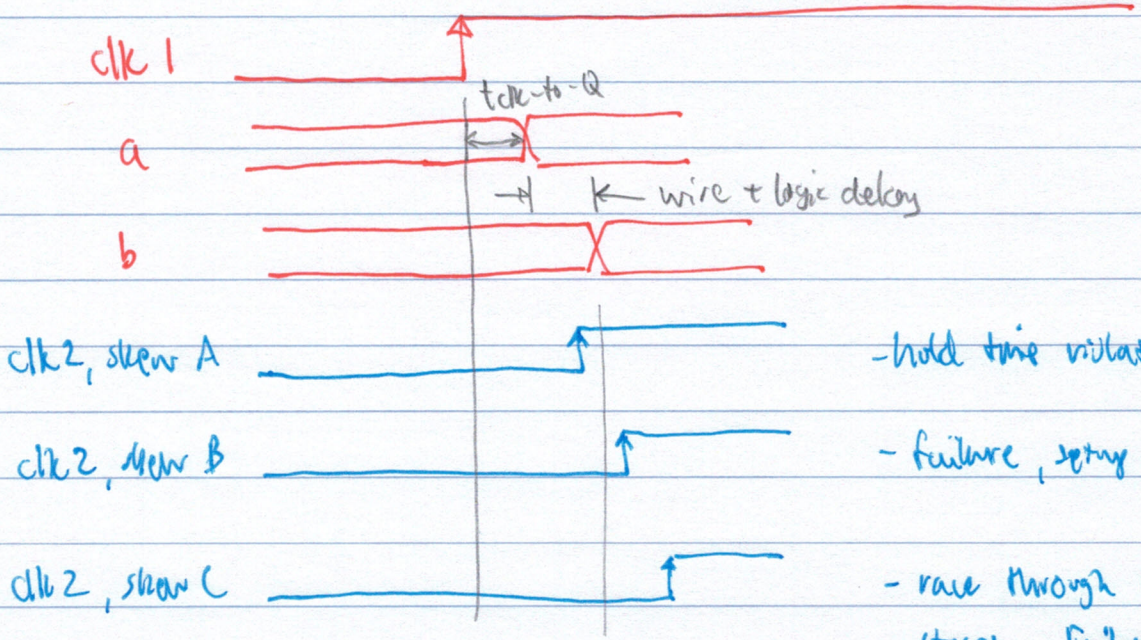
$$t_{cycle_{available}} = \frac{1}{f_{clk}} - \left| t_{clk\text{-}skew_{worst\ case}} \right|$$

Requirement #2 (logic is not too fast)



- Enabled by clk skew
- Happens:

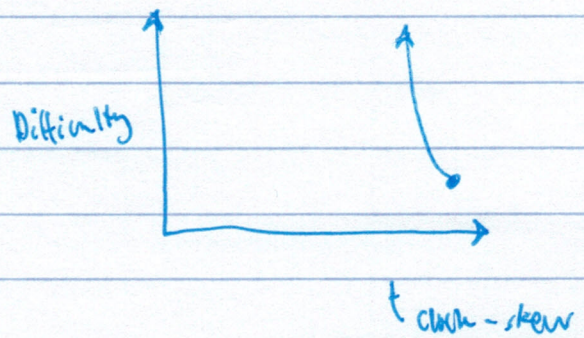




$$t_{clk\text{-}to\text{-}Q} + t_{logic_{min}} > t_{clk\text{-}skew} + t_{hold}$$ for correct operation

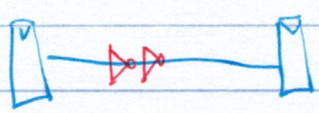
What if requirement is violated?

A) Design time



- Increase $t_{logic_{min}}$

Very easy!



Add two inv's (slow ones)

B) After chip is built

- can not change
- (no f_{clk} term)
- no fixes possible

∴ Hold time violations are very dangerous!

Hardware Description Languages

1) Verilog
   - industry
   - similar to C

2) VHDL
   - gov't

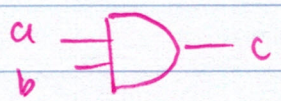
SW - code an algorithm

HDL - code HW

Verilog

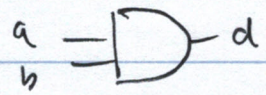
- Module

- wire - assign statement

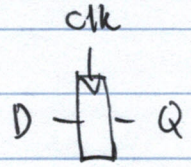



```
wire c;
assign c = a & c;
```

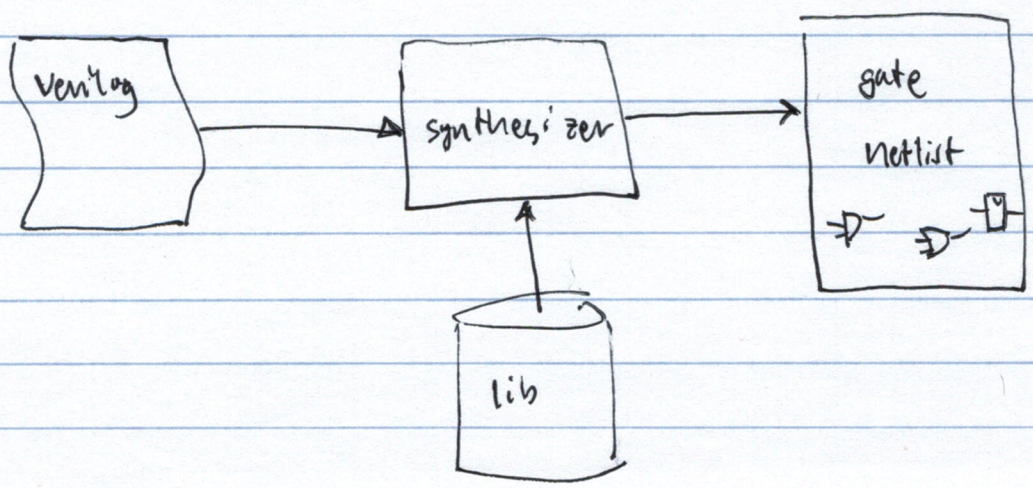
- reg - always block




```
reg d;                          <-- sensitivity list
always @(a or b) begin
    d = a & b;
end
```

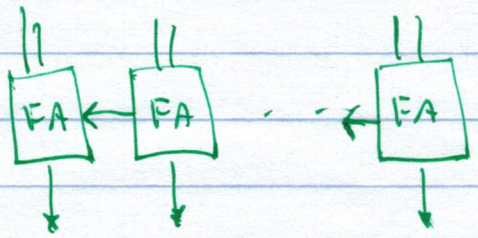



```
reg Q;
always @(posedge clk) begin
    Q = D;
end
```



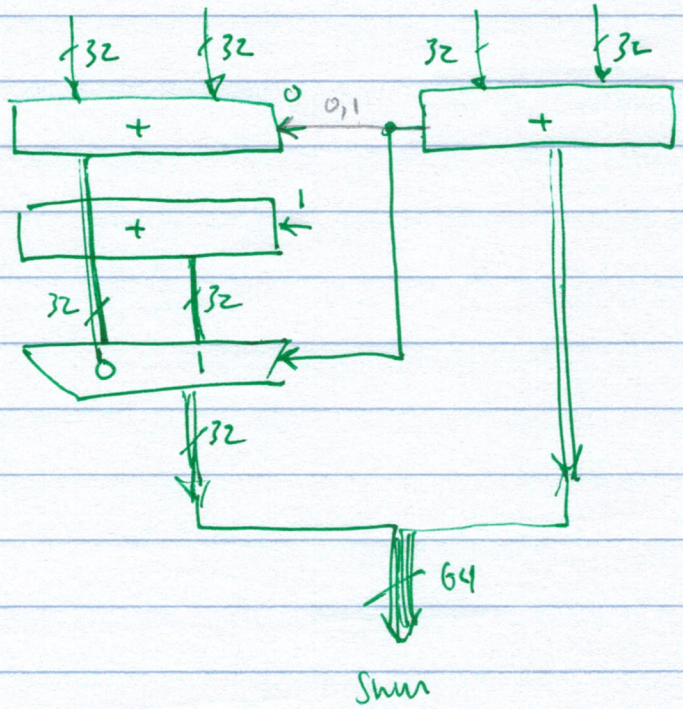
Adders

1) Ripple-Carry



- simplest
- slowest
- smallest

2) Carry-Select Adder

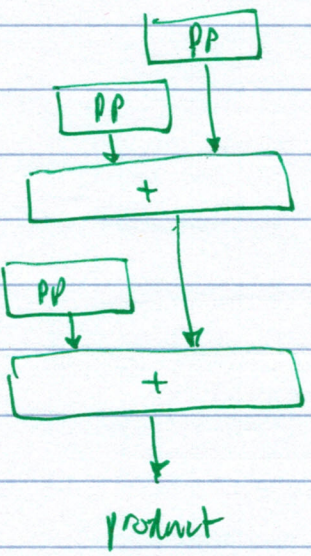


Multipliers

$$110_2 = 6_{10}$$

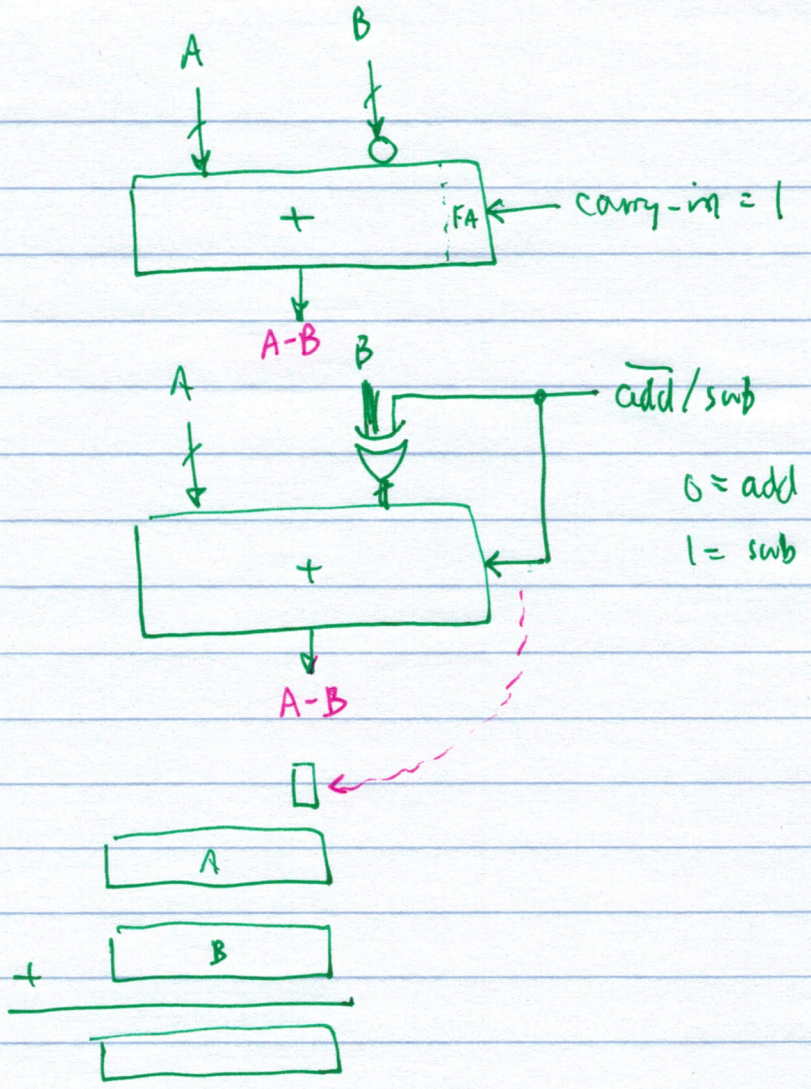
$$\times\ 101_2 = 5_{10}$$

$$\begin{array}{r} 110 \\ 000 \\ +\ 110 \\ \hline 11110 \end{array} = 30_{10} \checkmark$$



Subtractors

- Unsigned format $\quad x = 1-2$
- $A - B = A + (-B)$
  $= A + \overline{B} + 1$


A
B
+
FA
carry-in = 1
A-B
B
A
add/sub
0 = add
1 = sub
+
A-B
A
B
+