Low Level Digital Logic

Logic Gates

Performs Boolean Functions

- NOT 
- AND 
- OR 
- NOR 
- NAND 
- XOR 

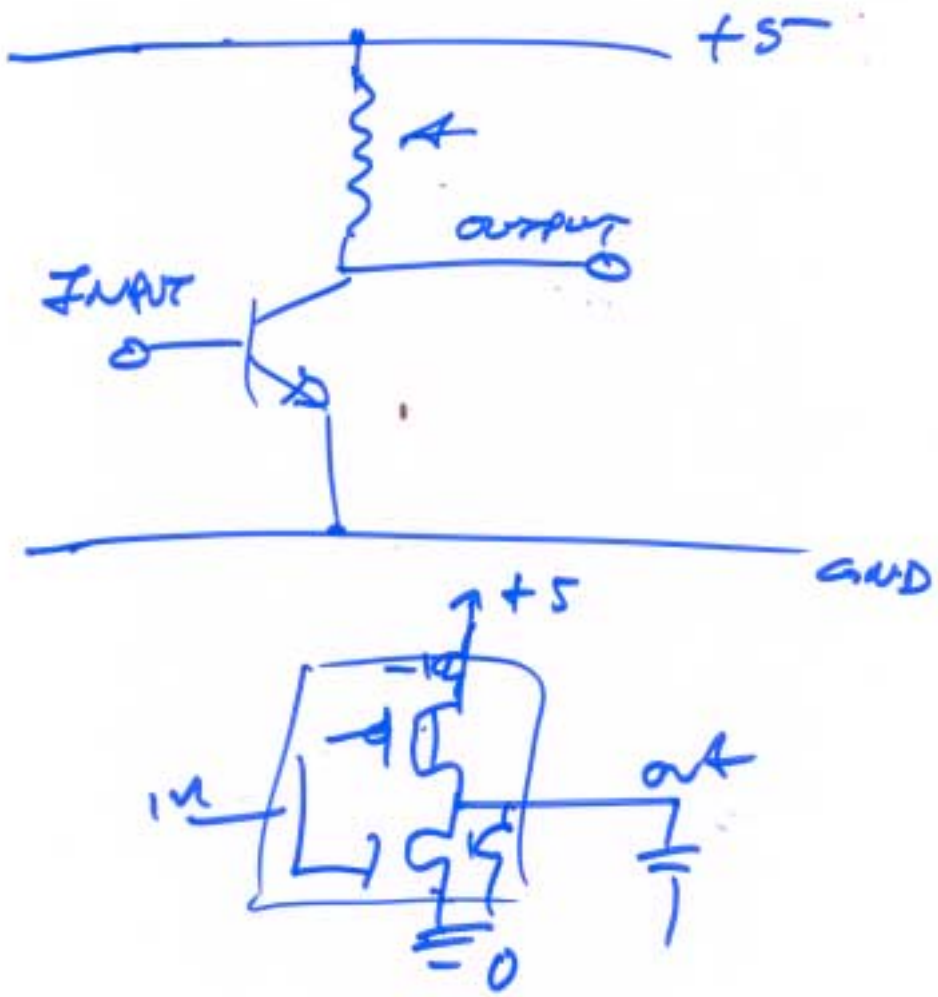
Logic Gates

Data Storage

- Flip Flop
- Latches
- Registers

Flow Control

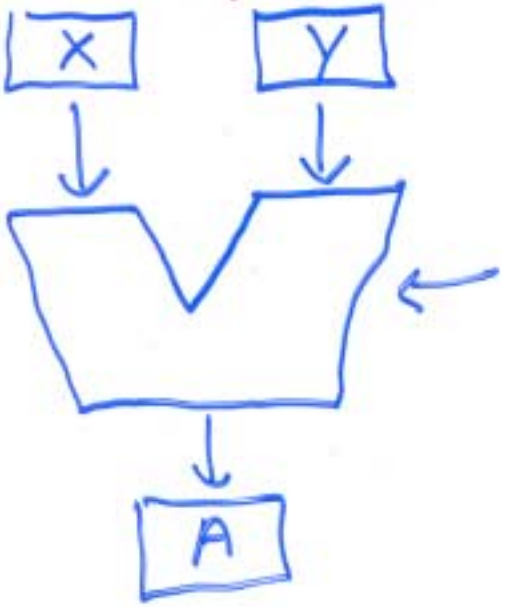
- Buffers
- Tri-State Buffers
- Multiplexor
- Demultiplexor


+5
R
OUTPUT
INPUT
GND
+5
IN
OUT
0

Arithemetic Logic

ALU -

Circuit designed to perform an arithemetic operation. Part of most programmable computers.



Operation Select

+, −, compare
AND, OR, Shift,
Address

Can be as simple or complex as required.

ALU Con't

- Partitioned arithemetic to increase performance
  - Multiple ALUs
- Pipeline to increase performance
- Vector ALU - perform same operation on multiple pieces of data.

Specialized Hardware

- Adders
- Multipliers
- Dividers

SEQUENTIAL LOGIC

MEMORY / STATE / REGISTERS

FEEDBACK

CLOCKED LOGIC

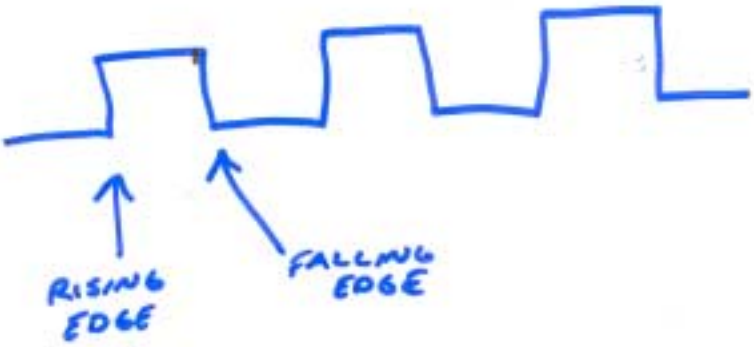
SYNCRONIZATION

CLOCKED LOGIC

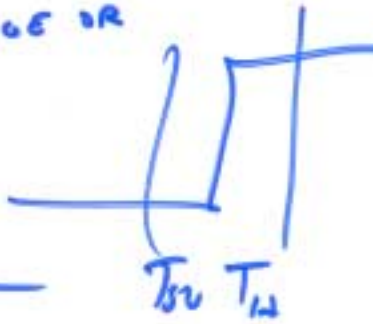
LOCKSTEP DESIGN
SIMPLIFIES THE DESIGN

↖ A "D" LATCH

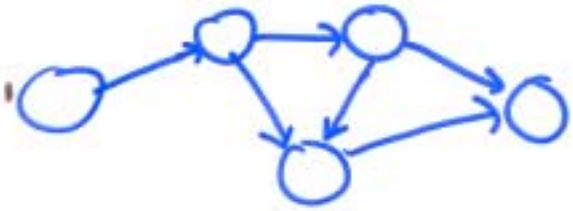
SYNCRONIZATION



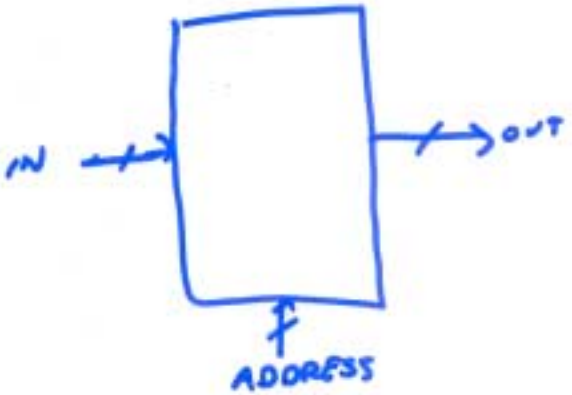
ALT ON EITHER EDGE OR ON/OFF



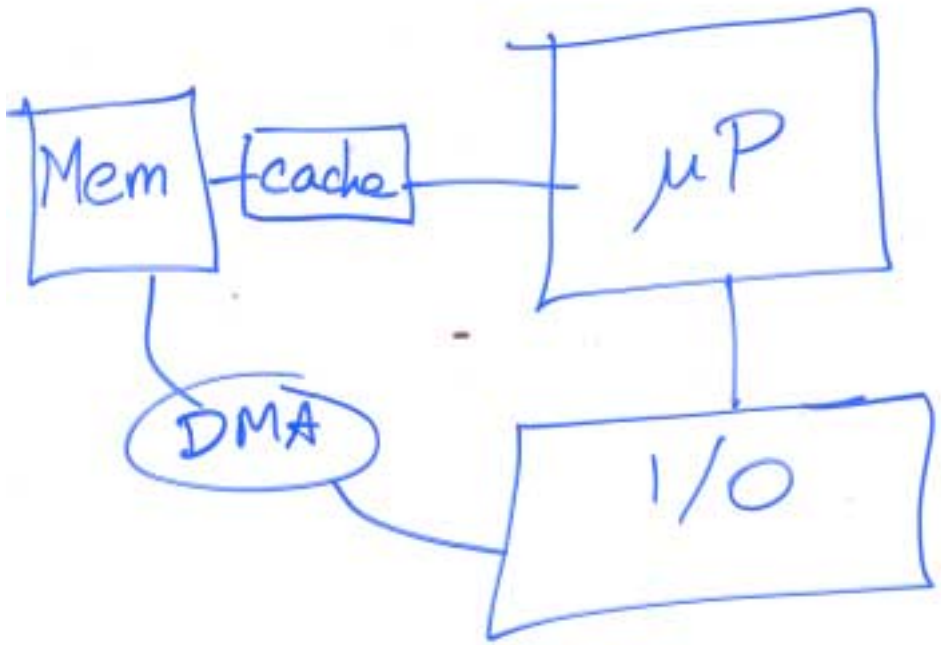
State Machine



Register Sets



MICROPROCESSORS



IMPORTANT ASPECTS OF DATAPATHS + CONTROL

- speeds – big differences in relative speed
- address sizes
- parallelism
- power consumption, materials
- error control
- cost

μP's VS. OTHER METHODS

- APPLICATIONS (COMPLEXITY)
- PROGRAMMABILITY + TOOLS
- COST
- RELIABILITY
- OFF-THE-SHELF VS. CUSTOM

Basic Digital Design Review

Simple Error Detection code generator: Sum of Bits

PACKET = <DATA> <BITS>
[7:0] [3:0]

Software Solution

CODE

```
{
PACKET = BITS(DATA);
SEND(PACKET);
}
```

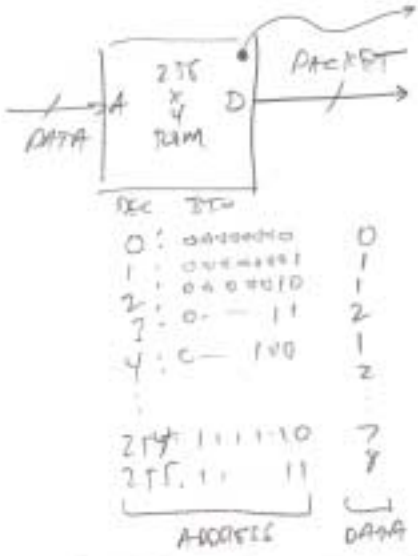
```
packet BITS(unsigned DATA) {
    packet.DATA = DATA;
    bits = 0;
    while (DATA) {
        if (DATA & 0x1) bits++;
        DATA = DATA >> 1;
    }
    Packet.bits = bits;
    return (Packet);
}
```

HW Solution

CODE

SEND (DATA)

What does the HW TTTS look like?



1k bit ROM – cost

1 level – performance

↓

function of one too!

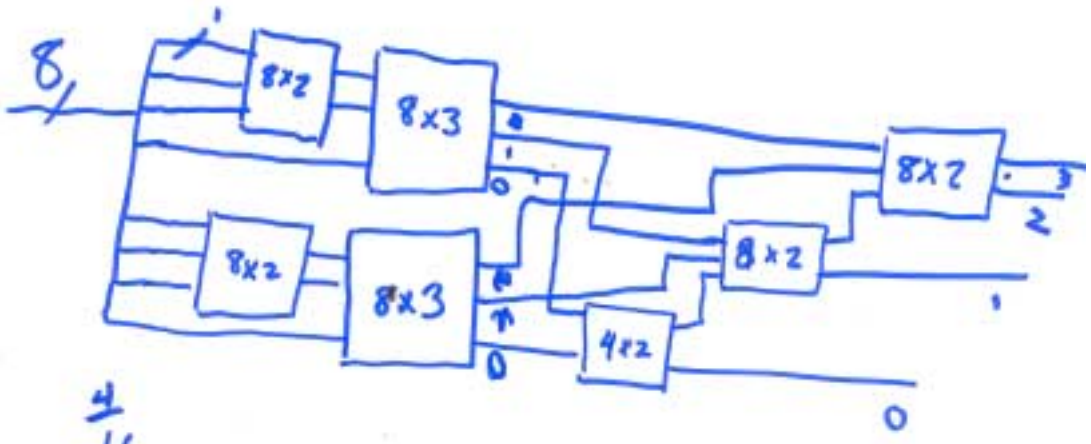
Design challenge – using only NOR gate – who has a clue?

2 min cost Gaps

2 min level Gaps < 500 bit

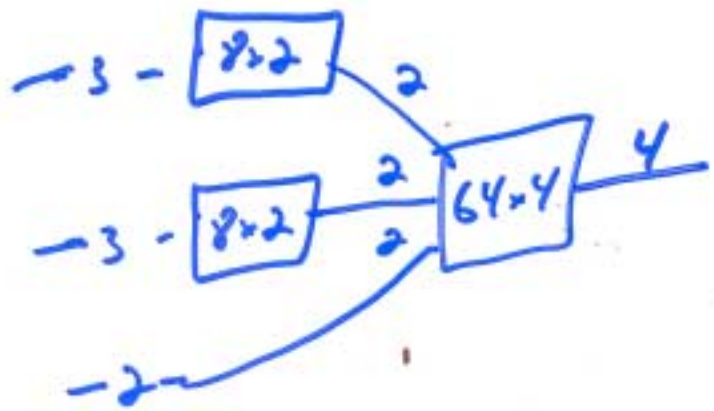
- Total cost
- Total performance

LOW COST



4
16
16
24
24
16
16
8
---
120 bits

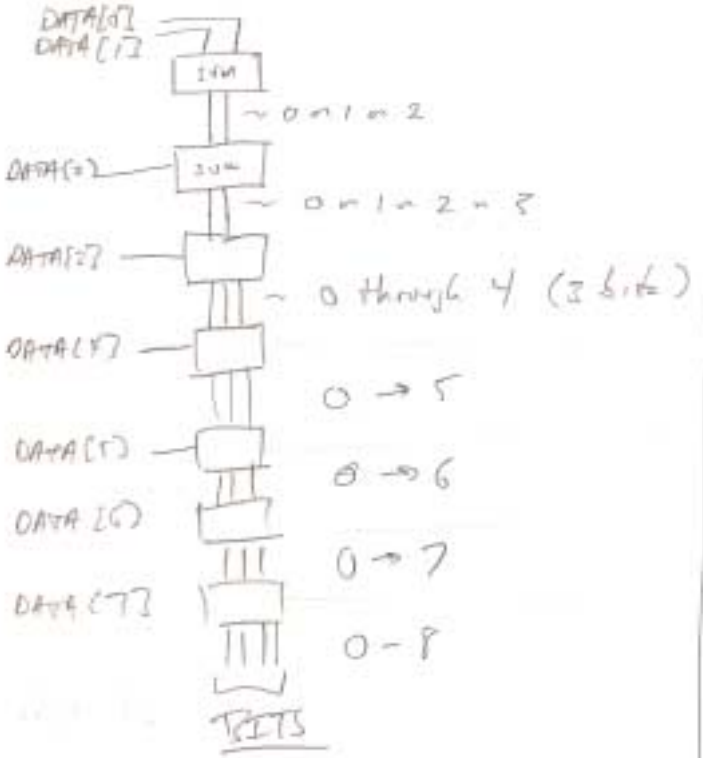
0	0	→ 0	0
0	1	→ 0	1
1	0	→ 0	1
1	1	→ 1	0



$16 + 16 + 256 = 288$

My Solutions

Min cost serial solution (Feel free for inspiration)



	Prom bits
	$2^2 \times 2 = 8$
	$2^3 \times 2 = 16$
	$2^3 \times 3 = 24$
	$2^4 \times 3 = 48$
	48
	48
	$2^4 \times 4$ 64
	270 bits

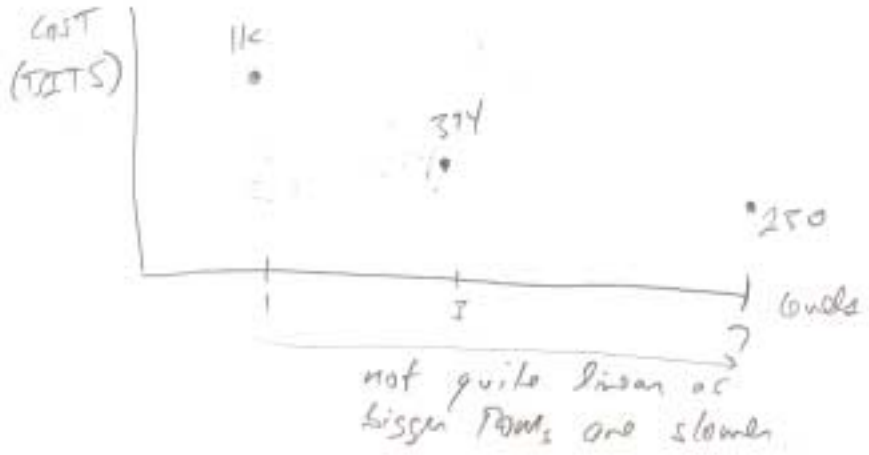
total it up cost = 270

Ref = 7 levels (small/lin) (linear)

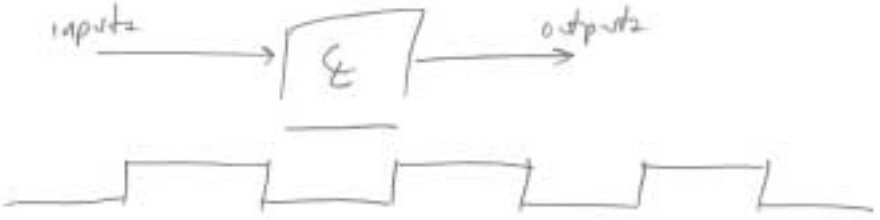
Compromise solution Timing Tree

COST = 374

Rel = 3 bits ($\log_2$)



Timing



clock has no physical effect on our circuit but we use it to mark time:

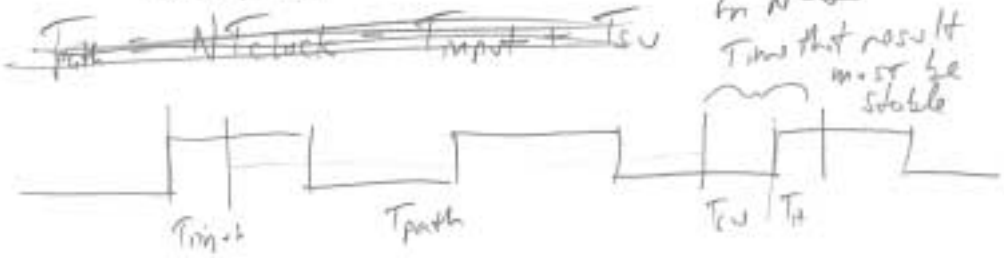
1. In what clock cycle does something happen
2. when in clock cycle does something happen

for combinational logic:

given input availability relative to rising edge:

specify output availability by cycle such that following is satisfied for each path:

~~$T_{path} = NT_{clock} - T_{input} + T_{su}$~~



EXAMPLE N=2

SLOW CASES

$T_{input} + T_{path} + T_{su} < NT_{clock}$

FAST CASES

$T_{input} + T_{path} > T_H$

Functional Analysis

Cycle-0 as rising edge when DATA_in ≠ DATA
→ clear TSTTS
→ load new DATA
→ load TMP for shifting

Cycle-1
if (TMP(0))
Bits = Bits + 1;
TMP = TMP >> 1

(PACKET IS STABLE AFTER N = 8 cycles
Cycle-N - same as cycle-1

System will halt reset when DATA changes

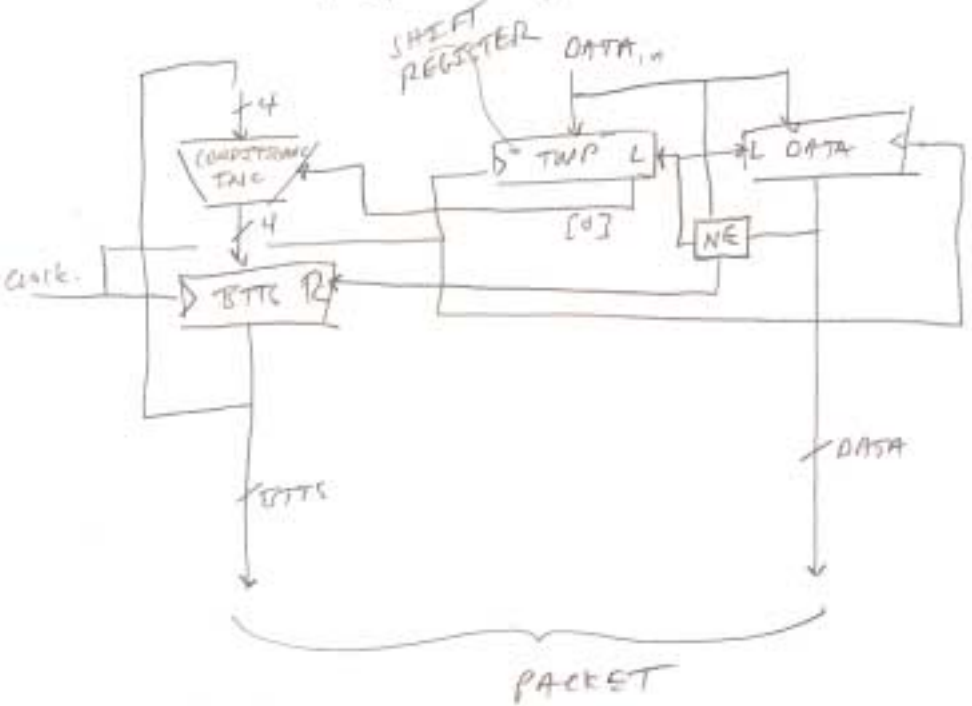
Timing N = 8, make sure that
Packet is stable on last rising edge (tmp = 00000001)
and check "internal timing" to make
sure inputs to registers are stable at
the ~~time~~ every rising edge.

Sequential Implementation

what is the point? HW Reuse (SW)!

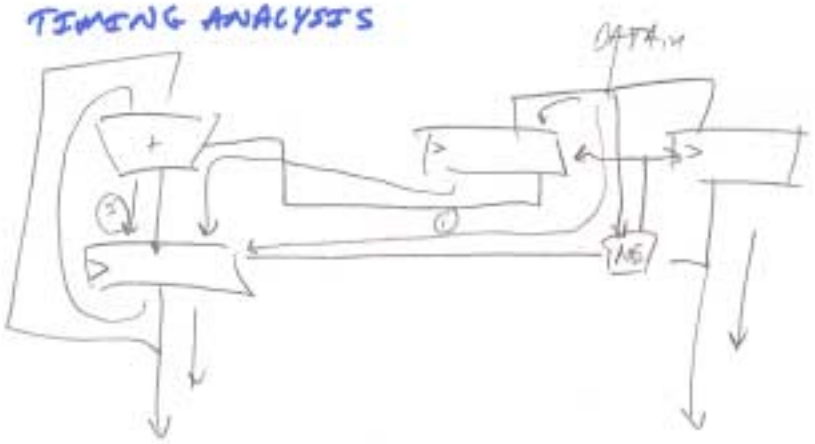
recall serial version

→ maybe we need just one of these!



Behaves exactly like combinational logic

TIMING ANALYSIS



clock internal paths:

① DATAin → SOFT BITS

$T_{DATAin} + T_{INE} + T_{RSU} < T_{clock}$

pre-specified! $T_{DATAin} + T_{INE} > T_{HOLD}$

② $T_{DATAin} + T_{SU} < T_{clock}$ and

$T_{DATAin} > T_H$

③ $T_p + T_{inc} + T_{SU} < T_{clock}$ (from TS_i to to TS_i+1)

no HOLD EQUATION

④ From TMP to Bits

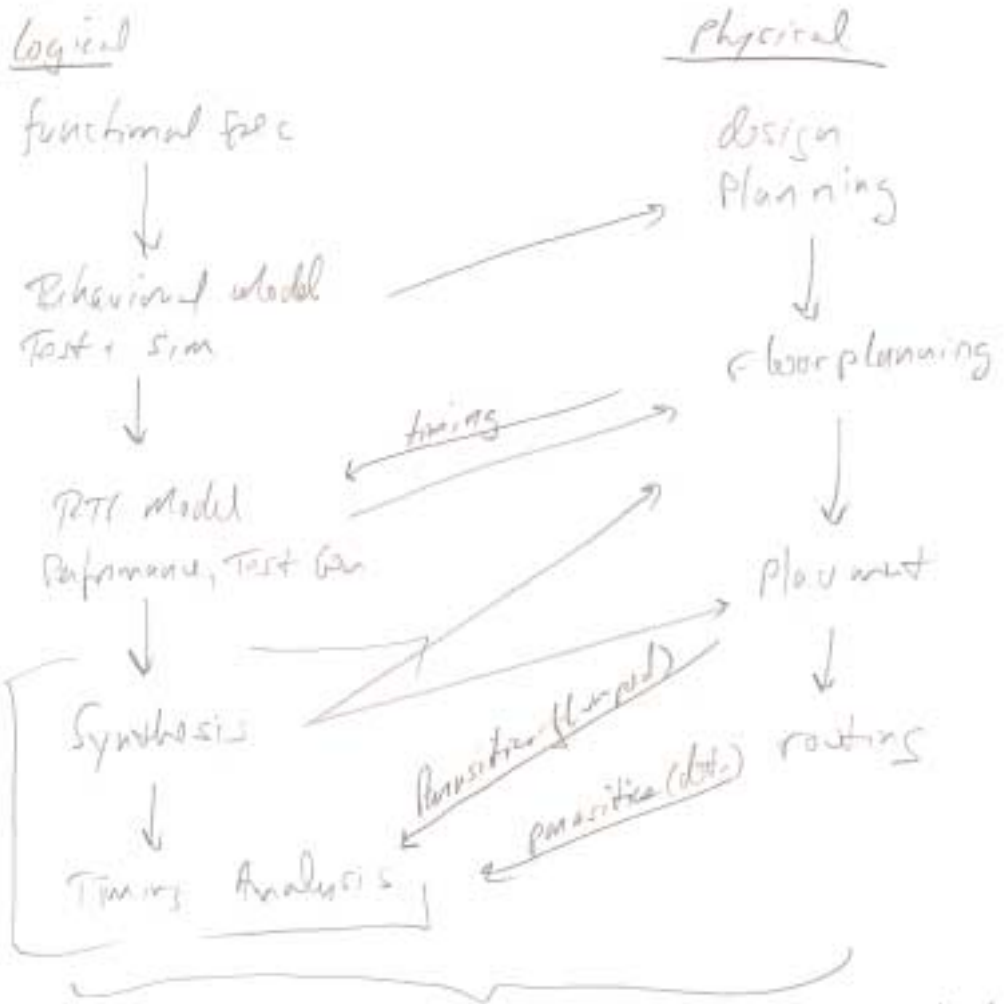
$T_p + T_{inc}^{*} + T_{SU} < T_{clock}$

⑤ Registers to Outputs

$T_p + T_o < T_{clock}$ output timing constraint

$T_p > T_{OH}$ Hold constraint

Typical Digital Design Flow



Tools/Details are App/Tech dependent