CMOS Transistors

NMOS G ⊣ D/S — GOOD 0, BAD 1

- $V_{GS} < V_{TH}$ → transistor OFF
- $V_G > V_{TN}$ → NMOS on
  - ↳ I proportional to V_{DS}

PMOS G ⊣ S/D — GOOD 1, BAD 0

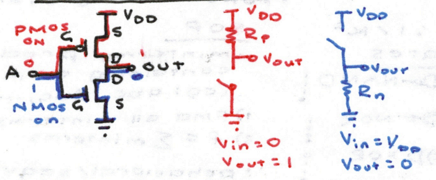
- $V_{GS} > V_{TH}$ → tr

- Symmetrical devices
- NMOS: drain at higher voltage
- PMOS: source at higher voltage

ON/OFF Model

$|V_{GS}|$; $|V_{GS}| < |V_T|$; R_{ON} ; $|V_{GS}| \geq |V_T|$

CMOS Inverter



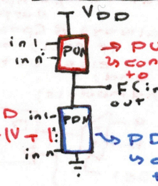
Basic Circuitry

$R = \frac{\rho L}{tW}$

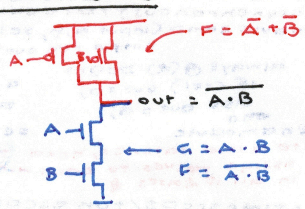
$R_{series} = R_1 + R_2 + R_3 + \dots + R_N$

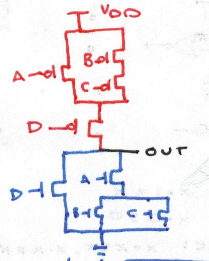
$R_{parallel} = R_1 \| R_2 \| \cdots \| R_N$

Pull-Up & Down Networks



- → PUN = F
  - ↳ conduction to VDD
- ↳ PDN = $\bar{F}$ = G
  - ↳ conduction to GND





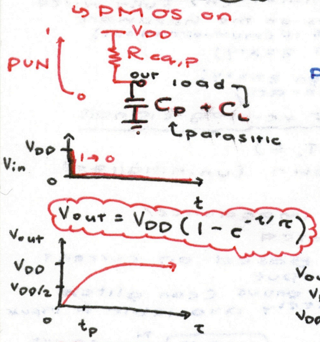
Inverter Delays

in → out, $C_L + C_P$

- delay in inverter chains minimized when each stage has the same fanout

Low-to-high

↳ PMOS on

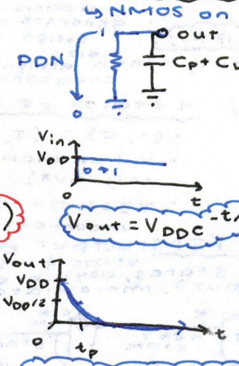


$V_{out} = V_{DD}(1 - e^{-t/\tau})$

$\tau = R_{eq,P}(C_P + C_L)$

High-to-Low

↳ NMOS on



$V_{out} = V_{DD} e^{-t/\tau}$

$\tau = R_{eq,n}(C_P + C_L)$

$t_p = \ln(2)\,\tau = 0.7\tau$

$t_p = \ln 2\, R_{eq}(\gamma C_{in} + C_L) = \ln 2\, R_{eq} C_{in}(\gamma + C_L/C_{in})$

$t_p = \tau_{inv}(1 + f)$

- ↳ $\tau_{inv} = \ln(2) R_{eq} C_{in}$ ← indep of transistor sizes
- ↳ fanout $f = C_L / C_{in}$

Normalized Delay to τ_{inv}

$D(inv) = t_p / \tau_{inv} = 1 + f$ (parasitic delay, effort delay)

- inverter delay: $t_{p,inv} = \gamma + FO$
- ~~delay expression~~ ~~$\sum_{i=1}^{N} \tau_{inv}(p_i + g_i f_i)$~~

Capacitances

- C_{in}: input capacitance
  - ↳ largely set by C_g (gate capacitance) $C_g \sim WL$
  - ↳ inverter: $C_{in,inv} = C_{g,pmos} + C_{g,nmos}$
  - ↳ $W \to 2W \Rightarrow C_{in} \to 2C_{in}$
- C_P: parasitic capacitance
  - ↳ largely set by C_d (drain capacitance) $C_d \sim W$ (drain area/perimeter)
  - ↳ inverter: $C_{p,inv} = C_{d,pmos} + C_{d,nmos}$
  - ↳ $W \to 2W \Rightarrow 2C_P$
- $C_d = \gamma C_g$ ($\gamma \approx 1$)
  - ↳ inverter: $C_P = \gamma C_{in}$

Inverter Sizing

- doubling W ($W \to 2W$)
  - → $C_{in} \to 2C_{in}$
  - → $C_P \to 2C_P$
  - → $R \to \frac{1}{2}R$
  - ↳ $I \to 2I$

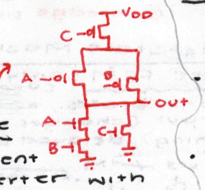
Inverter Chains

in1 → 1 → in2 → 2 → … inN → N → out, C_L

- delay … minimi… when … stage h… same f…
- Path Fanout (F)
  - $F = \frac{C_L}{C_{in,1}} = \prod_{i=1}^{N} f_i$
- Stage Fanout (f)
  - $f = \sqrt[N]{F} = \sqrt[N]{C_L / C_{in,1}}$
- Minimum path delay (D)
  - $D = Nf + N = N\sqrt[N]{F} + N = N(p + g$…
- size of each inverter stage (…
  - $C_{in,i} = \frac{C_{L,i}}{f_i}$
- overall (not min) path delay (…
  - $D = \sum(1 + f_i) = N + \sum f_i$ (fanout; parasitic delay)

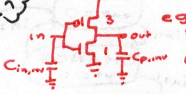
Logical Effort

$LE_i = \frac{C_g}{C_{inv}}$



How to calculate

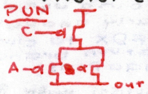
① Draw equivalent inverter (inverter with the same resistance)
- ↳ $R = \frac{1}{W}$ (want equivalent PDN & PUN resistance)



eg: $C_g = C$
- $R_{on,n} = R$, $R_{on,p} = 3R$
- $W_n = 1$, $W_p = 3$ results in PDN & PUN of R
- $C_{in,inv} = C_{g,pmos} + C_{g,nmos} = 1+3 = 4C$

② Find $R_{eq}, C_{in,inv}$
- $R_{eq} = R$
- $C_{in,inv} = C_{g,p} + C_{g,n} = 4C$

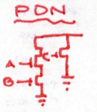
③ Size gate to have same output current as minimum size inverter (from ①)
- ↳ size for the worst-case. consider the PDN & PUN seperately
- → worst case for PUN: transistors in series
- → worst case for PDN: transistors in parallel



- the worst case is either A or B being on
- → want resistance to equal R, for the whole path
- ↳ know all these transistors have a resistance of $R_n = 3R$

$\frac{3R}{W_A} + \frac{3R}{W_C} = R$

$W_A = W_C = 6 = W_B$ (WLOG)



- worst case: either A & B or C being on
- $W_C = 1$ (bc we want it to equal R)

$\frac{R}{W_A} + \frac{R}{W_B} = R$

$W_A = W_B = 2$

④ $LE_{gate} = \frac{R_{eq,gate}}{R_{eq,inv}} \cdot \frac{C_{in,gate}}{C_{in,inv}}$
- ↳ if you size the gate to have same resistance as the inverter, simplifies to
- $LE_{gate} = \frac{C_{in,gate}}{C_{in,inv}}$

$C_{in,gate} = C_{gate}(W_{n,gate} + W_{p,gate})$

- $C_{in,inv} = C_g(3+1) = 4C$
- $C_{in,A} = C(6+2) = 8C = C_{in,B}$
- $C_{in,C} = C(6+1) = 7C$
- $LE_{,A} = 8C/4C = 2 = LE_{,B}$
- $LE_{,C} = 7C/4C = 7/4$

⑤ $P = \frac{R_{eq,gate}}{R_{eq,inv}} \cdot \frac{C_{p,gate}}{C_{p,inv}}$ (=1 if normalised)

$C_{p,gate} = \gamma C_g(W_{n,B} + W_{n,C} + W_{p,A} + W_{p,C})$

$C_{p,inv} = \gamma C_g(W_n + W_p) = 4\gamma C$

$C_{p,gate} = C_{gate}(W_{n,touching\text{-}out} + W_{p,touching\text{-}out})$; $P = 15/4$ … $= 15\gamma C$

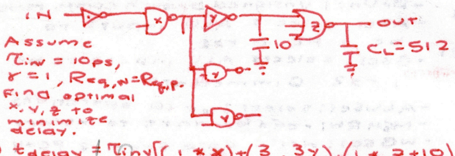
⑥ Total D… ($t_{p,tot}$)

$t_{p,tot} = t_{p,inv}$ … $= \tau_{inv}(F + f_{inv})$ … $+ \tau_{inv}(P_{gate}$…

$f_{gate} = \frac{C_L}{C_{in}}$

Branch Effort

- delay expression: $\sum_{i=1}^{N} \tau_{inv}(p_i + g_i f_i)$ (parasitic delay; logical effort)



Assume $\tau_{inv} = 10ps$, $\gamma = 1$, $R_{eq,n} = R_{eq,p}$. Find optimal x, y, z to minimize delay.

logical efforts of common gates:	
inverter:	1
NAND2:	4/3
NOR2:	5/3
NOR3:	7/3
NAND3:	5/3

Parastics	
inverter:	1
NAND2:	4/2
NOR2:	4/2
NOR3:	6/2
NAND3:	6/2

① $t_{delay} = \tau_{inv}\left[(1 * x) + \left(\frac{3}{2} \cdot \frac{3y}{x}\right) + \left(1 * \frac{z+10}{y}\right) + \left(\frac{4}{2} \cdot \frac{512}{z}\right) + \sum_{i=1}^{N} P_i\right]$

② take partial derivates of t_{delay}. set equal to 0 & solve for each

$\frac{\partial t_{delay}}{\partial x} = 1 - \frac{9y}{2x^2} = 0$, $\frac{\partial t_{delay}}{\partial y} = \frac{9}{2x} - \frac{z+10}{y^2} = 0$

$\frac{\partial t_{delay}}{\partial z} = \frac{1}{y} - \frac{1024}{z^2} = 0$ → solve system

$x = 8.55$, $y = 16.25$, $z = 129.01$

Branch factor (b_i): $b_i = \frac{\sum on + \sum off}{\sum on\text{-}path}$

Stage Effort

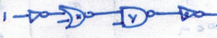
Stage Effort of Stage i (SE_i)

$SE_i = b_i f_i (LE_i)$ (fanout; logical effort; branching effort)

Multi-Stage Networks

- Delay $= \sum_{i=1}^{n}(P_i + LE_i * FO_i)$
- stage effort: $SE_i = LE_i * FO_i$
- Path Fanout Effort: $FO_{path} = C_{load}/C_{in}$
- Path logical effort: $LE_{path} = LE_1 \times LE_2 \times \dots \times LE_N$
- Path Effort: $PE = LE_{path} \times FO_{path}$
- Best stage effort: $\sqrt[N]{PE} = SE_*$
- minimum path delay: $N \cdot SE_* + P$
- best effective fanout of each stage: $FO_i = \frac{SE_*}{LE_i}$

Gate sizing exa…



- $LE_1 = 1$, $FO_1 = \frac{x}{1}$
- $LE_2 = \frac{5}{3}$, $FO_2 = \frac{y}{x}$
- $LE_3 = 3/2$, $FO_3 = z/y$
- $LE_4 = 1$, $FO_4 = 36/z$

$PE = \prod LE_i \cdot FO_i = 1 \cdot \frac{x}{1} \cdot \frac{3}{2} \cdot \frac{y}{x} \cdot \frac{3}{2} \cdot \frac{z}{y} \cdot 1 \cdot \frac{36}{z}$…

$SE_* = \sqrt[4]{81} = 3$ (optimal)

$D = N \cdot SE_* + P = 4 \cdot 3 + (1 + \frac{4}{2} + \frac{4}{2} + 1) =$ …

$C_{in} = LE * \frac{C_{out}}{SE_*}$

$z = 1 \times \frac{36}{3} = 12$, $y = \frac{3}{2} \times \frac{z}{3} = \frac{z}{2} = 6$ …

$x = \frac{3}{2} \times \frac{y}{3} = \frac{y}{2} = 3$

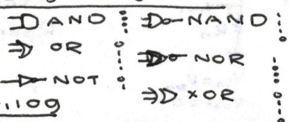
Dennard Scaling

- device dim (t_{ox}, L, W), current, voltage, capacitance ($\epsilon A/t$), delay time per circuit (VC/I) : 1/K
- doping concentration (N_a) : K
- power dissipation per circuit (VI) : $1/K^2$
- power density (VI/A) : 1

power/performance

- Power $P = \frac{1}{2}CV^2 f$
- Performance (f)
- Power density = VI/A

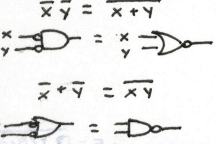
Logical gates



Boolean Algebra

DeMorgan's law:

$\overline{xy} = \overline{x}+\overline{y}$



$\overline{x+y} = \overline{x}\,\overline{y}$

identities

$x+0=x$ $\quad x\cdot 1=x$

$x+1=1$ $\quad x\cdot 0=0$

- idempotence: $x+x=x$, $x\cdot x=x$
- complements: $x+\overline{x}=1$ $\quad x\cdot\overline{x}=0$
- Commutative: $x+y=y+x$ $\quad x\cdot y=y\cdot x$
- Associative: $x+y+z=((x+y)+z)$
- Distributive: $x\cdot(y+z)=x\cdot y+x\cdot z$
- Absorptive: $x+x\cdot y = x\cdot(1+y)=x$
  $x\cdot(x+y)=(x+0)\cdot(x+y)=x+(0\cdot y)=x$

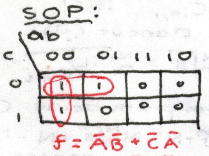
$\overline{F(x_1,x_2,\dots,x_n,0,1,+,\cdot)}^D = F(x_1,x_2,\dots,x_n,0,1,+,\cdot)$

- Simplification:
  $x\cdot y + x\cdot\overline{y} = x$
  $(x+y)\cdot(x+\overline{y}) = x$
  $x\cdot(x+y)=x$
  $x+x\cdot y = x$

Karnaugh maps

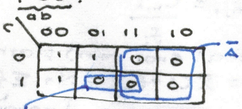
a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

SOP:



$f = \overline{A}\overline{B} + \overline{C}\overline{A} = \overline{A}(\overline{B}+\overline{C})$

POS:

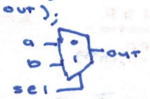


$f = (\overline{A})(\overline{B}+\overline{C})$

Verilog

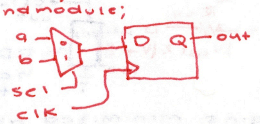
- structural/combinational (asynchronous)/blocking

```
module comb (input a,b, sel,
        output reg out);
  always @(*) begin
    if (sel) out = b;
    else out = a;
  end
endmodule
```



- to avoid latches, need to assign values to each wire in all if cases
- behavioral/sequential (synchronous)/nonblocking

```
module seq (input a,b,sel,clk)
        output reg out);
  always @ (posedge clk) begin
    if (sel) out <= b;
    else out <= a;
  end;
endmodule;
```



- timesteps/tim escale
  `timescale (simulation timestep)/(simulation time resolution)
  #n_units_of_time = #n_simulation_timesteps
  - time resolution: smallest possible time unit
  - timescale 1ns/10ps ⇒ #1 = 1ns
    time resolution ≥ 0.01 = 10 ps
- initial begin blocks <condition> ? <if T> : <if F>;
  - only executes once, runs sequentially, (=) blocking assign
  - #(n) : delay by n timesteps
  - $display("string: prints formatted string");
  - $finish; // ends simulation
- simulation constructs

```
reg clk;
initial clk=0;
always #(10) clk <= ~clk;
repeat (10) @(posedge clk);
@(posedge signal);
```

}creates clk with 20 timestep period; // wait for 10 posedges; // wait for 1 signal

SOP

- minterms: product terms containing all input combos (eg: abc, $\overline{a}$bc, ...) (output=1)
1) Find all minterms where output=1
2) $f = \sum$ minterms

POS

- maxterm: sum terms involving all input combos (eg ($a+b+c$), ($\overline{a}+b+c$), ...) where output = 0
1) Find all maxterms
2) $\overline{F} = \sum$ maxterms
3) $F = \overline{\sum \text{maxterms}}$

2D regs/memory

reg [31:0] mem [7:0];
   (#bits)   (#elements)

mem[2]; // 3rd 32-b element
mem[5][7:0]; // 6th elements' lowest Byte (8b)

Generate loops

```
genvar i;
generate for (i=0;i<n;i=i+1) begin
  full_adder u0(.a(a[i]),...,.k(k[i]...
end
endgenerate;
```

↳ instantiates (n) fully wired adders on the hardware

```
generate if (mult_option==1)
  assign z = x*y;
else
  assign z = x+y;
endgenerate;
```

4 states of verilog signals

- (1, 0) = (T, F)
- x = unknown (uninitialized reg/bus)
- z = high impedance/unconnected

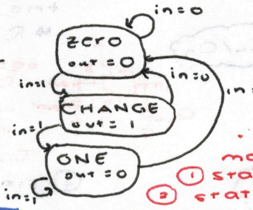
Finite-State Machines (FSM)

Moore FSM: only based on current state

- more states, synchronous output (changes based on clk timing), delayed by 1 cycle

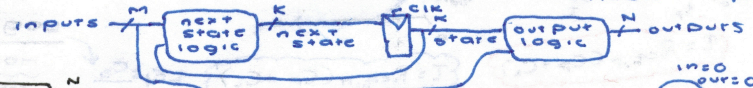
- Moore edge detector

input	curr state	next state	output
0	zero	zero	0
1	zero	change	0
0	change	zero	1
1	change	one	1
0	one	zero	0
1	one	one	0



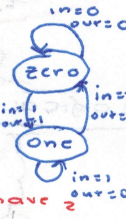
Mealy FSM: output based on current state & input

- less states, asynchronous (can glitch w/input), immediately available w input



- Mealy edge detector:

input	curr state	next state	output
0	zero	zero	0
1	zero	one	1
0	one	zero	0
1	one	one	0



- for FSMs, in general, we have 2 main sections in Verilog code
  1. state transition: sequential
  2. state/output logic: combinational

RISC-V

- R-type: <inst> rd, rs1, rs2
- I-type: <inst> rd, rs1, imm
  or <inst> rd, imm(rs1)
- S-type: <inst> rs1, imm(rs2)
- B-type: <inst> rs1, imm(rs2)
- J-type: jal rd, label
- U-type: <inst> rd, imm

I/S types:
- byte addressing
- byte-level granularity
- lower 4 bytes (little endian)

J: writes PC+4 to rd unless rd is x0
U: puts upper 20 bits of imm into rd

RISC-V Control Signals

- inputs:
  - instruction - read from instr memory
  - BrEq: 1 if inputs to branch comparator are equal
  - BrLT: 1 if input1 < input2
- outputs
  - PCSel: 0 ALU output, 1 PC+4
  - ImmSel: 1 of 5 types based on imm typ
  - RegWEN: 1 Enable writeback to registerFile
  - BrUn: 1 Unsigned branch comp. mode
  - ASel: selects ALU input1 to be 1 PC 0 rs1
  - BSel: selects ALU input2 to be 1 immediate 0 rs2
  - ALUSel: select 1 of 10 operations
  - MEMRW: write to memory
  - WBSel: write data from 2 PC+4 1 ALU 0 DataMemory to rd

RISC-V Hazards

- structural - 1 resource required by >1 instruction
  ↳ soln: add more hardware
- data: 1+ src register(s) not up-to-date when being used → soln: Forwarding/NOP
- control: for branch instructions, don't know early enough if branch is taken
  ↳ soln: guess then flush/fix if wrong. merge FDX to eliminate control hazards (bc they're caused by timing diff btwn branch comparison and PC use happens)

Control Logic Truth Table

inst[31:0]	Breq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	REN	WB
add	*	*	+4	*	*	reg	reg	add	read	1	ALU
sub	*	*	+4	*	*	reg	reg	sub	read	1	ALU
R-R op	*	*	+4	*	*	reg	reg	(op)	read	1	ALU
addi	*	*	+4	I	*	reg	imm	add	read	1	ALU
lw	*	*	+4	I	*	reg	imm	add	read	1	Mem
sw	*	*	+4	S	*	reg	imm	add	write	0	*
beq	0	*	+4	B	*	PC	imm	add	read	0	*
beq	1	*	ALU	B	*	PC	imm	add	read	0	*
bne	0	*	ALU	B	*	PC	imm	add	read	0	*
bne	1	*	+4	B	*	PC	imm	add	read	0	*
blt	*	1	ALU	B	0	PC	imm	add	read	0	*
bltu	*	1	ALU	B	1	PC	imm	add	read	0	*
jalr	*	*	ALU	I	*	reg	imm	add	read	1	PC+4
jal	*	*	ALU	J	*	PC	imm	add	read	1	PC+4
auipc	*	*	+4	U	*	PC	imm	add	read	1	ALU

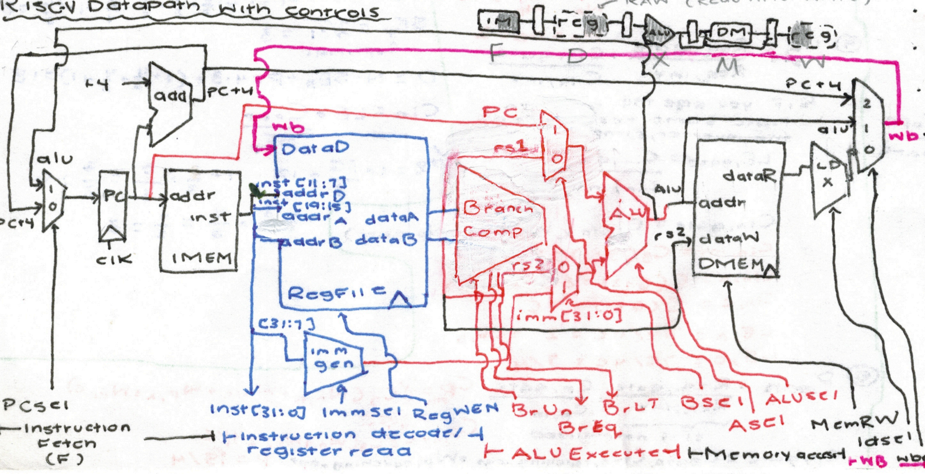
Pseudoinstruction

- beqz rs1 label
  ↳ beq rs1 x0 label
- bnez rs1 label
  ↳ bne rs1 x0 label
- j label (PC=rs1 + offset)
  ↳ jal x0 label
- jr rs1 (PC=rs1)
  ↳ jalr x0 rs1 0
- la rd label
  ↳ auipc, addi
- li rd imm
  ↳ lui (if necessary), addi
- mv rd rs1
  ↳ addi rd rs1 0
- neg rd rs1
  ↳ sub rd x0 rs1
- nop
  ↳ addi x0 x0 0
- not rd rs1
  ↳ xori rd rs1 -1
- ret (PC=ra)
  ↳ jalr x0 x1 0

Some nuance

- b-type instr. only store bits [12:1] of the immediate
  → bltu x2, x3, 5 (PC=0x40, x2<x3) goes to PC=0x44 (drops last bit of 5 during encoding)
- verilog ⟺ RISC-V
  - xor rd rs1 rs2
    ↳ wire[31:0] ALU_out = a[31:0] ^ b[31:0];
  - sll rd rs1 rs2
    ↳ ALU = a[31:0] << b[4:0]
  - srl rd rs1 rs2
    ↳ ALU = a[31:0] >> b[4:0];
  - sra rd rs1 rs2
    ↳ = $signed(a[31:0] >>> b[4:0]
- increase clk f by pipelining
  - total stall cycles = (#mispred)(cost)
  - stall cycle for branch misprediction is 3 cycles bc while we know if the branch is taken or not at the X stage, the target addr is computed at the M stage

RISCV Datapath with controls



Single-Cycle Datapath Equations

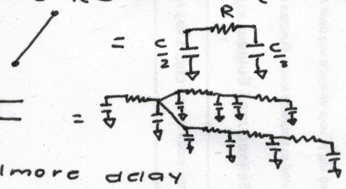
- critical path/clk cycle/clk period:
  $t_{clk} \geq t_{clk\text{-}to\text{-}q} + t_{setup} + t_{largest\ combinational\ block}$
- hold time:
  $t_{hold} \leq t_{clk\text{-}to\text{-}q} + t_{shortest\ CL}$
- minimum clk cycle: (min poss. val for t_{clk})
- max clk freq:
  $f = 1/t_{min,cc}$
- throughput = (# operations)/time
- latency = (# stages in DP)·(t_{clk})

Branching

- branch factor $b_i = \frac{\Sigma_{on-path} + \Sigma_{off\_path}}{\Sigma_{on-path}}$
- Path branching effort: $B = \Pi b_i$
- Path Effort: $PE = B \cdot FO_{path} \cdot LE_{path} = B\left(\frac{C_{out}}{C_{in}}\right)(\Pi LE_i)$
- Best stage effort $SE_* = \sqrt[N]{PE} = b_i \cdot fo_i \cdot LE_i$
- Path delay $D = N \cdot SE_* + P$

Elmore Delay

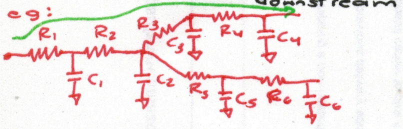
- wire RC model (π model)



- Elmore delay
  - → start at $\tau = 0$
  - → τ at each resistor,

$\tau_+ = R \cdot \Sigma C_{downstream}$

eg:

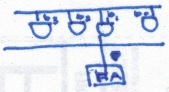


$\tau = R_1(C_1 + C_2 + C_3 + C_4 + C_5 + C_6)$
$+ R_2(C_2 + C_3 + C_4 + C_5 + C_6)$
$+ R_3(C_3 + C_4)$
$+ R_4(C_4)$

Multipliers

Parallel Array Multiplier

a3 a2 a1 a0
× b3 b2 b1 b0
+ a0b0
+ a0b1
+ . . . a0b2
+ . . . a0b3
P7 ... P1 P0



Booth Recoding

$(B_{K+1,K})(A) = 0A \to 0$
$= 1A \to A$
$= 2A \to 4A - 2A$
$= 3A \to 4A - A$

B_{K+1}	B_K	B_{K-1}	action
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

0111 A
× 1010 B
- 01110

Energy

$E_c = \int_0^\infty I(t)V(t)dt = \int_0^\infty C\frac{dV(t)}{dt}V(t)\,dt$

$= C\int_0^{V_c} V(t)\,dV = \frac{1}{2}CV_c^2 \leftarrow$ Energy

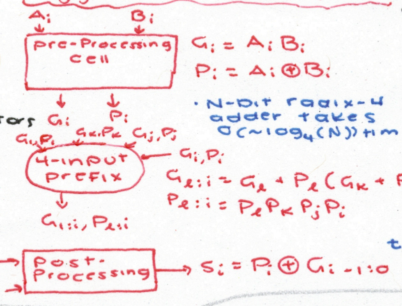
$P_c = \frac{E_c}{T} = \frac{1}{2}CV_c^2 f \leftarrow$ Power (watts)

$C = \varepsilon\frac{LW}{d} \leftarrow$ capacitance

- Energy per instruction:
  $E_{inst} = P_c \cdot (t_{inst})$
  $= P_c(CPI \cdot \frac{1}{f})$
- Energy for n instr:
  $E_n = n \cdot E_{inst}$
- Energy delay prod. for n instructions:
  $EDP = n \cdot E_n (CPI)(\frac{1}{f})$

- to achieve lowest power design for the minimum delay, additional inverters/buffers should be placed right before the load capacitor
- dynamic power $= P_{dynamic} = \alpha C V_{DD}^2 f$, $\alpha \in [0,1]$
- switching power - when both transistors are on ($V_{dd} - V_{thp} > V > V_{tn}$), ie, when it's switching from on → off or vice versa $P_{switching} = IV = V^2/R$
- leakage power (when $R_{off} < \infty$)

Kogge-Stone (radix 4) adder



$G_i = A_i B_i$
$P_i = A_i \oplus B_i$

- N-bit radix-4 adder takes $O(\sim\log_4(N))$ time

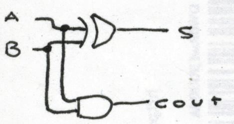
$G_{\ell:i} = G_\ell + P_\ell(G_K + P_K(G_j + P_j G_i))$
$P_{\ell:i} = P_\ell P_K P_j P_i$

$S_i = P_i \oplus G_{i-1:0}$

$t_{K-S\,r4} = t_{pre-processing} + 2t_{prefix} + t_{post-processing}$

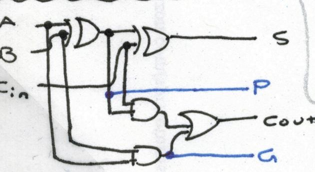
Adders

Half Adder

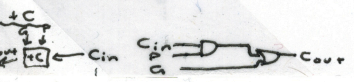


$S = A \oplus B$
$C_o = AB$

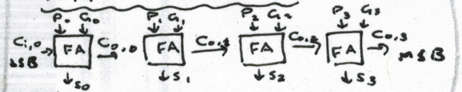
Full Adder



$S = A \oplus B \oplus C_i$
$C_o = AB + BC_i + AC_i$
$P = A \oplus B$
$G = AB$

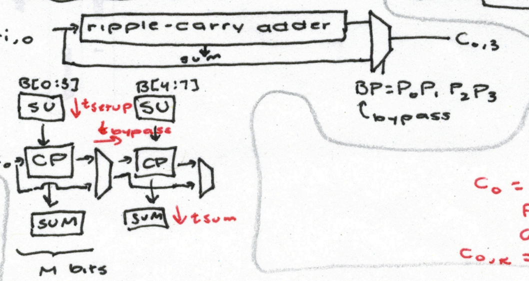


Ripple Carry Adder



$t_{adder} = (N-1)t_{carry} + t_{sum}$ (FA, $C_i \to C_o$; FA, $C_i \to S$)

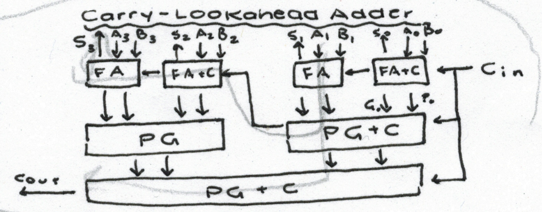
Carry-bypass/skip adder

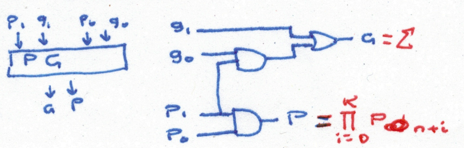


$BP = P_0P_1P_2P_3$
t_{bypass}

$t_{carry\text{-}bp\text{-}add} = t_{setup} + (M-1)t_{carry} + \left(\frac{N}{M-1}\right)t_{bypass} + (M-1)t_{carry} + t_{sum}$

Carry-Lookahead Adder





$P = \prod_{i=0}^{K} P_{n+i}$

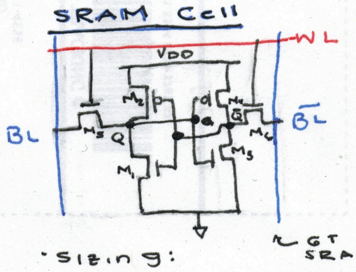
$C_o = AB + BC_i + A_i = G + PC_i$
$P = P_1P_2P_3 \cdots P_N$
$G = g_N + P_N g_{N-1} + \cdots + (P_1P_2 \cdots P_N)g_1$
$C_{o,K} = G_K + P_K C_{o,K-1}$

Flip Flops

Timing

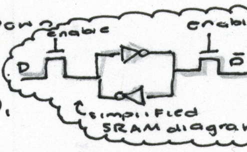
- minimum cycle time set by longest logic path:
  $T > t_{clk-to-q} + t_{logic,max} + t_{setup}$
- setup slack $= T - (t_{clk-to-q} + t_{logic,max} + t_{setup})$
- hold time constraint:
  $t_{hold} < t_{clk-q} + t_{logic,min}$
- hold slack $= t_{clk-to-q} + t_{logic,min} - t_{hold}$
- when we have jitter & skew:
  - minimum cycle time:
    $T > t_{c-q} + t_{logic,max} + t_{su} - t_{sk} + t_j$
  - hold time constraint:
    $t_{hold} + t_{sk} + t_j < t_{c-q} + t_{logic,min}$
  - hold slack:
    $t_{clk-to-q} + t_{logic,min} - (t_{hold} + t_{skew})$
  - setup slack:
    $T - (t_{clk-q} + t_{logic,max} + t_{setup}) + t_{skew}$

SRAM Cell



- write: BL & $\overline{BL}$ are inverted
- read: BL & $\overline{BL}$ are V_{DD}
- wordline enables read/write access for a row
- → GT SRAM can only support 1 read & 1 write port

- Sizing:
  Pull Up : Access : Pull-Down
  1 : 2 : 3
  $(\frac{W}{L})_2 : (\frac{W}{L})_5 : (\frac{W}{L})_1$



- writeability: $(\frac{W}{L})_2 < (\frac{W}{L})_5$ necessary
- read stability: $(\frac{W}{L})_5 < (\frac{W}{L})_1$ necessary
- the bitline that is pulled low is the one involved in flipping the cell state during a write operation
- in a GT SRAM techniques like adjusting voltages of wordlines, bit lines, or the latch pair that improve read stability, hurt writeability be they're coupled together (unlike BT cells)
- SRAM cell leakage degrades read access over time

Memory Implementation

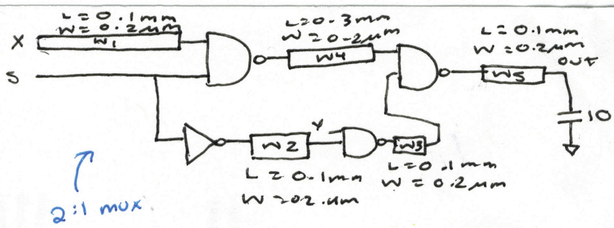
- SRAM cell: height $h_{\mu m}$ & width $w_{\mu m}$ (horizontal WL, vertical BL). wire capacitance of C_w. Supply voltage of V_s. Has n rows, m columns
  - → dimensions: (n rows)(h μm)
  - → wordline capacitance:
    $C_{WL} =$

pg. 6, 16, 19, 21

Find minimum unsigned entry in an asynch read RAM by iterating through one entry per clk cycle.

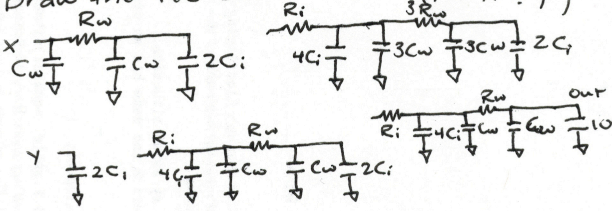
```
reg [7:0] ram [0:15];
reg [4:0] addr; //(2^4=16=# entries in ram)
reg [7:0] minimum; //each ram entry is 8 bits wide
always @(posedge clk) begin
    if (rst)
        addr <=0;
    else
        addr <= addr+1;
end
always @ (posedge clk) begin
    if (rst)
        minimum<= ram[0];
    else
        minimum<= (ram[addr]<minmum)?ram[addr]:minimum;
end
```

- If the regfile is asynch read, synch write, we can remove the explicit registers btwn the mem access & WB stages & maintain the same fcnality
- For B-format instr. if we assume branch always taken, we need extra hardware in the F stage to calculate the new address



- W_1 has resistance R_w & parasitic capacitance $2C_w$
- Inverters have input cap C_i, parasitic cap $2C_i$ & output res R_i
- NAND - input $2C_i$, parasitic $4C_i$, output R_i

Draw the RC switch model (for X & Y)



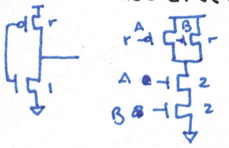
cycle ALU, 2 N-bit inputs, M distinct operations, you can simulate the ALU at a speed of C Hz (C clk cyc./sec). How long does it take to exhaustively test the ALU? $(2^N \cdot 2^N \cdot M)/C$ seconds

Behavioral description of a decoder

```
wire [7:0] addr;
wire [255:0] dec;
assign dec = 256'd1 << addr;
```

1 bit decoder logic

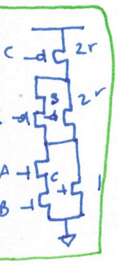
```
wire addr;
wire [1:0] dec;
assign dec[0] = !addr;
assign dec[1] = addr;
```



$LE_{,A} = \frac{r+2}{r+1}$

$LE_{,B} = \frac{r+2}{r+1}$

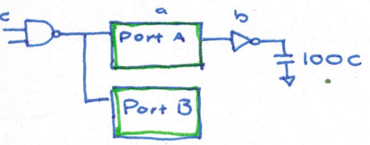
← NAND2



$LE_{,A} = \frac{2r+2}{r+1} = 2$

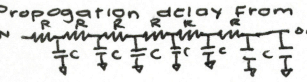
$LE_{,B} = \frac{2r+2}{r+1} = 2$

$LE_{,C} = \frac{2r+1}{r+1}$



$PE = FO_{path} \cdot B \cdot LE_{path} = (100)(2)(\frac{r+2}{r+1} \cdot 2 \cdot 1)$
$= 400(\frac{r+2}{r+1})$

Propogation delay from input to output?



$t_p = \ln 2(R \cdot 7C + \dots + R2C)$
$= (\ln 2)(27RC)$

Suppose you can add as many buffers to path to minimize path delay. How many should you add?

Previously:

PE=216 Path Delay=24 P=6 N=3

Now: $D = N\sqrt[N]{PE} + P = N\sqrt[N]{216} + 6 + 2x$

buffers = 0 → N=3 D=24

=1 → N=5 $D = 5\sqrt[5]{216} + 6 + 2 \approx 22.65$ ← (bc 2 inverters)

=2 → N=7 $D = 7\sqrt[7]{216} + 6 + 4 = 25.09$

↳ add 1 buffer
→ place it before the largest load C_L

Physical Array Organization

SRAM Array. 1024 entries of 8 bits. Each cell is 0.12µm high & 0.5µm wide (where WL horizontal, BL vertical). Wire capacitance of 0.2fF/µm & supply voltage of 0.9V.

If you build an array of 1024 rows & 8 columns, how much energy is used to drive the WL?

$E_{WL} = C_{WL} \cdot V^2 = (8\text{ cells} \cdot \frac{0.5\mu m}{1\text{ cell}} \cdot \frac{0.2fF}{1\mu m})(0.9V)^2 = 0.65fJ$

(one WL turns on)

If you build the same array as (a part (a)), how much energy is discharged from the bitlines (assuming the WL is left on for a long time)?

$E_{bl} = C_{bl} \cdot V^2 = (8\text{ columns} \cdot \frac{1024\text{ cells}}{1\text{ column}} \cdot \frac{0.12\mu m}{1\text{ cell}} \cdot \frac{0.2fF}{1\mu m})(0.9V)^2$

$= 159fJ$ (all bitlines discharge)

Now say we have 256 rows, 32 columns. What E_{WL}?

$E_{WL} = C_{WL} \cdot V^2 = (32\text{ cells} \times \frac{0.5\mu m}{1\text{ cell}} \cdot \frac{0.2fF}{1\mu m})(0.9V)^2 = 2.5fJ$

& E_{bl} = ?

$E_{bl} = (32\text{ cols} \times \frac{256\text{ cells}}{1\text{ col}} \cdot \frac{0.12\mu m}{1\text{ cell}} \cdot \frac{0.2fF}{1\mu m})(0.9V)^2 = 159fJ$

CSA array multiplier

↳ critical path (for 8x8 array with 7 bit ripple-carry adder)

$= t_{partial\text{-}product} + 8t_{FA} + 7t_{FA}$

Caches

- direct mapped cache. 8KB size, 64B blocks. memory addresses 32 bits
  - → offset bits: 64 B blocks = 2^6 bytes
    ↳ 6 offset bits
  - → index bits: $ size = 8 KB = 2^{13} B
    13 − 6 = 7 index bits
  - → tag bits: 32 − 6 − 7 = 19 tag bits

SCRAM

byte-addressable memory; 64 words of storage. 1:1 aspect ratio.

- → memory capacity in bits: 64 × 64 = 4096 = 4 K bits
- → rows, cols = $\sqrt{4096}$ = 64
- → mem capacity in bytes: 64 × 8 = 512 B
- → number of address bits: $\log_2(512) = 9$ bits
- → # column decode bits:
  ↳ each row has 64 columns = 64 b = 8 B
  ↳ $\log_2(8) = 3$ column decode bits
- → # row decode bits = # address bits − column decode bits
  = 9 − 3 = 6