CL | Boolean Algebra Laws:

$X\cdot(Y+Z) = XY + XZ$ | $X+YZ = (X+Y)(X+Z)$

$\overline{XY} = \bar{X} + \bar{Y}$ = ¬(x·y) | $\overline{X+Y} = \bar{X}\cdot\bar{Y}$ = ¬(x+y)

$X(X+Y) = X$ | $X + XY = X$

$(X+Y)(X+\bar{Y}) = X$ | $XY + X\bar{Y} = X$

$(X+\bar{Y})Y = XY$ | $X\bar{Y} + Y = X + Y$

$(X+Y)(\bar{X}+Z)(Y+Z) = (X+Y)(\bar{X}+Z)$

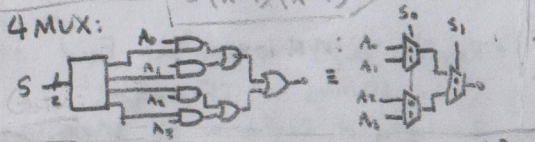
$XY + \bar{X}Z + YZ = XY + \bar{X}Z$

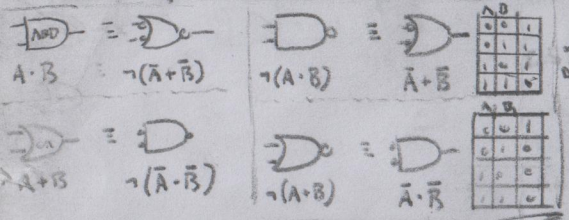
XOR: $X \oplus Y = (X+\bar{Y})(\bar{X}+Y) = \bar{X}Y + X\bar{Y}$

XNOR: $X \odot Y = (X+Y)(\overline{XY}) = \bar{X}\bar{Y} + XY$

$= (X+Y)(\bar{X}+\bar{Y})$

4 MUX:





Energy/Power: Joules = Watts × Time

- Energy consumption determines battery life (could be lighter/cheaper)
- Power consumption leads to more heat possibly limits Moore scaling

Ex | Dissipates P = 50W

Needs to last 9hrs — 1620 kJ

$\Rightarrow E = 50 \times 9 = 450$ Wh = 50 × 32 Wh

Needs to charge in 2hrs

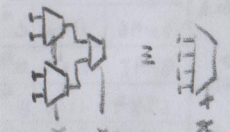
What must charger supply?

$\Rightarrow P = 50W + 1620/2hr = 275$ W

LUT | Boolean Func lookup table

- Width = n: 2^n inputs, 2^{2^n} mappings
- 2^n FFs, mux tree of $2^{n+1}-1$
  - ↳ Delay ∝ O(n)

LUT2:



- Truth table is unique
- Bool. expr. is easy to manipulate
- Gate repr. is closest to implementation

Canonical Forms (conjunctive & disjunctive)

1] Sum-of-Prod. OR minterms for when output is true

- minterm = product of all inputs

2] Prod-of-Sum. AND maxterms when 0

- maxterm = sum of all inputs
  - ↳ NOT'd for $\bar{F}$ expression

Ex | $F = \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c} + abc$

$\bar{F} = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c}$

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

POS: $F = (a+b+c)(a+b+\bar{c})(a+\bar{b}+c)$

$\bar{F} = (a+\bar{b}+\bar{c})(\bar{a}+b+c)(\bar{a}+b+\bar{c})$

$(\bar{a}+\bar{b}+c)(\bar{a}+b+\bar{c})$

K-Maps For simplification

Fewer gates ⇒ less area, less wires

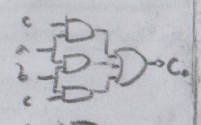
$F = (\bar{b}+c+d)(\bar{a}+c+\bar{d})(b+c+\bar{d})_{CW}$

ab \ cd	00	01	11	10
00	1	•	•	1
01	•	1	0	0
11	1	1	1	1
10	1	1	1	1

Full Adder Cell:

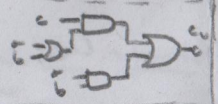
$C_1 = ac + ab + bc$

$r = a \oplus b \oplus c$



$C_0 = c(a+b) + ab$

$r = (a \oplus b) \oplus c$



FSMs | Represent sequential logic; output conditioned on all prior inputs

FSMs distill prior inputs into state

- Write TT with inputs & current state on LHS and next state encoding RHS:
- Can then write as SoP:

$S0' = \bar{S}_1\bar{S}_0X + \bar{S}_1S_0\bar{X}$

$S1' = \bar{S}_1S_0X + S_1\bar{S}_0X$

S1	S0	X	S1'	S0'
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0

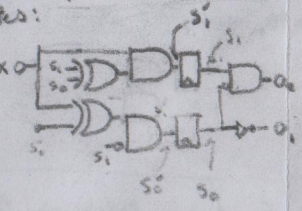
- IF we have outputs, write TT w/curr state encoding LHS and out on RHS:
- Reduced Form can be translated to gates:

S1	S0	O1	O0	O_1 =
0	0	1	1	$\bar{S}_1\bar{S}_0 + S_1\bar{S}_0$
0	1	0	0	$\bar{S}_0$
1	0	1	0	$O_0 = \bar{S}_1\bar{S}_0$

$S_1' = X(S_0 \oplus S_1)$

$S_0' = \bar{S}_1(S_0 \oplus X)$

$O_1 = \bar{S}_0$; $O_0 = \bar{S}_1\bar{S}_0$



Timing:

- Set-up: Time before clk st. input is stable
- Hold: Time after [" " " "]
- clk-to-Q: Time for signal to go through register
  - ↳ Vary w/capacitive load, input slope, curr Ω

c2q + longest path + setup ≤ period

c2q + shortest " + setup ≥ hold

↳ o/w input will change too fast

Critical Paths in 5-stage: $t_{clk} \geq \max(\cdot)$

IF: [PC c2q] + [IMEM read] + [R. setup]

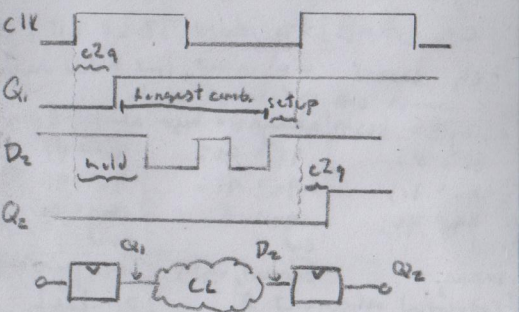
ID: [R. c2q] + [RF read] + [ ]

EX: ["] + [MUX] + [ALU] + [ ]

MEM: ["] + [DMEM read] + [ ]

WB: ["] + [MUX] + [RF setup]

- Moore: out dep. only on current state
- Mealy: " " " " inputs & "
  - ↳ State always depends on "
- Timing: Moore out follow clk edge; while Mealy out rises w/input rise, and out. falls sync. w/clk edge
- Moore is delayed, and may have more states
- Mealy async outputs immediately available but may not be stable long enough



Dennard Scaling:

- Voltage (& currents) ∝ transistor dim.
- Delay ≃ $C \cdot V/I_{on}$ ≃ linear
- Power ≃ $C \cdot V^2/Delay$ ≃ square
- In theory, const. power density

Feat scale 0.8 ⇒ K = $\frac{1}{0.8} = \frac{5}{4} = 1.25$

w/height, voltage, current, capacitance, delay: 1/K

Clk Freq, doping concentration: K

area, power per circuit: $1/K^2$

instr	BrEq	BrLt	PCsel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
(R-R op)	x	x	+4	x	x	Reg	Reg	[op]	R	1	ALU
addi	"	"	"	I	"	"	"	Add	"	"	"
lw	x	x	+4	I	x	Reg	imm	Add	R	1	Mem
sw	x	x	+4	S	x	Reg	imm	Add	W	0	↔ x
beq	0	x	+4	B	x	PC	imm	Add	R	0	↔ x
	1	x	ALU	"	"	"	"	"	"	"	"
bne	0	x	+4	"	"	"	"	"	"	"	"
	1	x	ALU	"	"	"	"	"	"	"	"
blt	x	1	ALU	"	0	"	"	"	"	"	"
bltu	x	1	"	"	1	"	"	"	"	"	"
jalr	x	x	ALU	I	x	Reg	imm	Add	R	1	PC+4
jal	x	x	ALU	J	x	PC	imm	↓	↓	↓	PC+4
auipc	x	x	+4	U	x	PC	imm				ALU

- Store word: Mem[reg[rs1] + imm] = reg[rs2]
- IF bltu xA xB 5 executed and PC = x40, new PC = x40 + ([5>>1]<<1) = x44

Iron Law: $\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruct.}} \times \frac{\text{Time}}{\text{Cycle}}$

1st term determined by ISA, compiler, source code
2nd by microarchitecture, ISA complexity
3rd by [" "], critical path, base code.

Hazards:
Pipelining n-stages reduces critical path, allowing higher clk Freq. ∝ n, yet thruput may scale ∝ $\frac{1}{n}$ due to dependencies/hazards

1] Structural. Required resource (hardware) is already in-use
- Resolved by duplicating/partitioning hardware
- Generally avoided in RISC-V

2] Data. Dependency between instructions
- Must wait/stall/spin 'til prior inst. reads/writes
- Resolved w/ forwarding (select val X stage ahead, rather than from reg)

3] Control. Flow of execution depends on prior instructions (i.e. branch)
- Branch determination in X-stage; after then we read correct instr. but stages must be flushed (no-ops) and next instr. begins fetch during branch wb
  - ↳ Bad pred: WB & IF (at label)
  - ↳ Correct ": Sequential greedy instrs okay

] Same with jump: when j is in WB, [label]'s instr is in IF stage

Pipelining Diagram Rules:
- Double-pump allows write sync. and read async (new val) in same cycle
  - ↳ Data dep: WB & ID simultaneously
- Forwarding:
  - WB → ALU: WB & EX simultaneously
    - Branches must still stall
  - ALU reg → ALU: M & EX simultaneously
- RegFile in IF ⇒ data dep fixed by stalling in IF stage; similarly if RegFile in EX stage.

Verilog] - Defaults to unsigned types
- reg [31:0] mem [7:0] has 8, 32-bit elts
  e.g. mem[2] is third 32-bit elt
- reg cannot be input; can't follow assign
  " always LHS in 'always' block

```
always@* begin  | always@* begin | always@* begin
  q1 = d;       |   q3 = q2;     |   q1 <= d;
  q2 = q1;      |   q2 = q1;     |   q2 <= q1;
  q3 = q2;      |   q1 = d;      |   q3 <= q2;
end             | end            | end
```

wrong [sequential assignment] when blocking	correct, ugly	proper parallel

- Latches take on value during whole high period of clk, not just on rising edge like flip-flops

Synthesis takes in source code (behavioral RTL) & constraints (i.e. clk freq, area). First, elaboration occurs, unrolling generates etc. before being mapped to generic logic gates. These are then converted to technology-specific gates determined by the PDK library and reports (timing, power, area) are generated. Gate-level verilog is our final output, with delays between gates/gate capacitance loads etc.

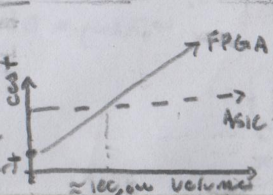
Place-and-Route takes in this netlist from synth. and constraints + PDK. Floorplanning first allocates area for IO ring, die area, power straps. Clock Tree Synthesis balances clk skew, fixes setup + hold timings. Assuming timing can be met, routing is legalized and parasitics/delays more accurately accounted for for final refinement

Implementations:

1) Full custom: Hand-drawn + optimized transistor layout
- High non-reoccurring engineering costs; time consuming layout iterations
- Common for analog; I/O, memory arrays (digital)

2) Standard Cells. Logic gates + "macros" automatically placed + routed
- Reduced NRE due to automation
- Typically 10x performance & 10x less power wrt FPGA

3) Gate Array. Pre-Fab. wafer, customizable routing of gates
- Shifts design + mask NRE costs to vendor
- Often include special circuits, RAMs/USB/PCIe
- Relatively low silicon efficiency ⇒ higher cost per part



2^n	n_2	n_x
1	0000	0
2	0001	1
4	0010	2
8	0011	3
16	0100	4
32	0101	5
64	0110	6
128	0111	7
256	1000	8
512	1001	9
1024	1010	A 10
2048	1011	B 11
4096	1100	C 12
8192	1101	D 13
16384	1110	E 14
32768	1111	F 15