Digital Logic Design

Curated undergraduate study track (not accredited, no external exam). Original explanations and practice only, following the reviewed course handout.

Bits, checked by exhaustion

- Two voltage levels, a handful of gates, and time.
- Every Boolean answer verified by evaluating all input rows.

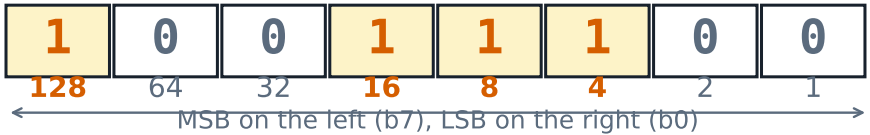
Number representations: weights, complement, overflow

1. **Convert** 156 = 128+16+8+4; hex 9C = 9*16+12.
2. **Negate** -5 in 8 bits: 00000101 -> 11111011.
3. **Overflow** 90 + 70 = 160 > 127; the 8-bit result reads -96: **overflow** by the sign rule.

Read bits by their weights

128 + 16 + 8 + 4 = 156

hex: 1001 1100 = 0x9C

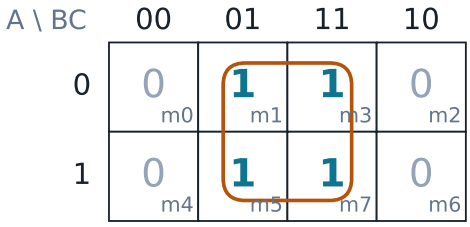


Weights 128 to 1 under each bit: the pattern 10011100 sums to 156, and grouping fours gives hex 0x9C.

Boolean algebra and Karnaugh maps: adjacency is simplification

1. **Odd rows** Minterms 1,3,5,7 all share only C, so F = C.
2. **De Morgan** (A+B')' = A · B - break the bar, change the sign.
3. **Check** Evaluate over all 8 rows; the map and the expression must agree everywhere.

Adjacency does the simplification



group of four: only C stays 1, so F = C

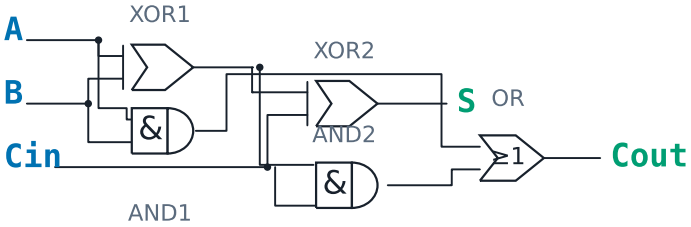
The odd minterms 1,3,5,7 form one four-cell group that keeps only C: F = C, verified on all 8 rows.

Combinational building blocks: route, decode, add

1. **Adder** At 1,1,1: S = 1, Cout = 1 - **binary** 11 for three ones in.
2. **Ripple** 1011 + 0110 stages ripple to 10001 = 17.
3. **Hazard** F = AB + A'C glitches when B = C = 1 and A changes; add BC.

Five gates, three inputs, two outputs

The full adder: five gates, three inputs, two outputs

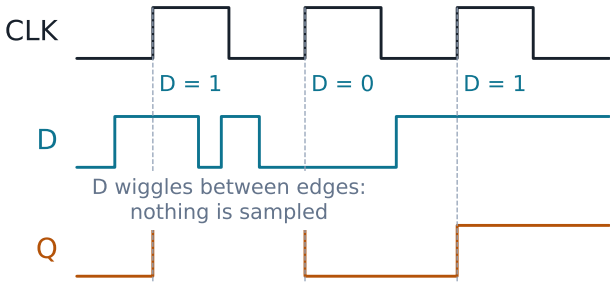


The **full adder**: XOR pairs make the sum, two ANDs and an OR make the carry - the most-reused circuit in computing.

Latches and flip-flops: level vs edge

1. **Sample** D reads 1, 0, 1 at three edges: Q reads 1, 0, 1 - wiggles between edges ignored.
2. **Toggle** J-K at (1,1) **toggles**: Q after (1,0),(1,1),(0,1),(1,1) is 1, 0, 0, 1.
3. **Design** Choose inputs per transition from the **excitation table**; the x entries are free minimisation room.

Q waits for the edge



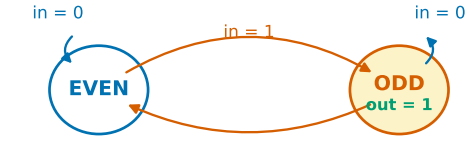
D wiggles freely between edges; Q copies it only at rising edges - 1, 0, 1 in the trace.

Finite state machines: behaviour to hardware

1. **Parity** EVEN/ODD states; Q+ = Q XOR in; trace 1,0,1,1 lands ODD on the fourth bit.
2. **Fall detect** Three states S0, S1, F; trace 1,1,0,0,1,0 outputs 0,0,1,0,0,1.
3. **Merge** States with equal outputs and equal next states are one state.

Two states, one flip-flop

The parity detector: two states, one flip-flop



next-state logic: $Q^+ = Q \oplus in$ — the machine is a parity bit

EVEN and ODD swap on input 1, loop on 0; output = 1 in ODD. The next-state logic is Q xor in.

Registers, counters and memory: n bits at a time

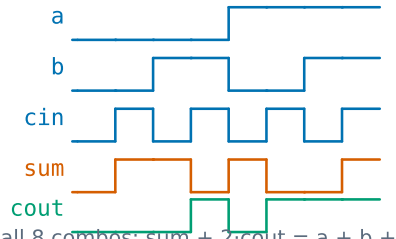
1. **Down-count** Bit 1 **toggles** when Q0 = 0, bit 2 when Q1Q0 = 00: 111,110,101,100,011,010,001, wrap.
2. **Shift** Inputs 1,1,0,1 clock four times: 1000, 1100, 0110, 1011.
3. **Memory** 8K x 8: 13 address lines (8K = 2^13), 8 data lines.

Description and proof together

One module, one exhaustive testbench

```
module full_adder(
  input a, b, cin,
  output sum, cout);
  assign sum =
      a ^ b ^ cin;
  assign cout =
      (a & b) |
      (cin & (a^b));
endmodule
```

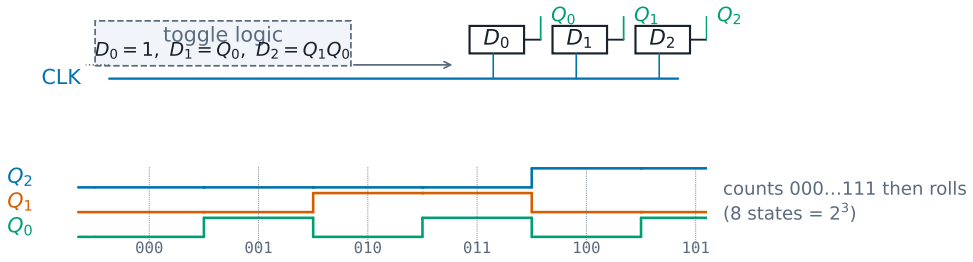
synthesizable module



all 8 combos: sum + 2·cout = a + b + cin ✓

Left: the full_adder module. Right: the **testbench** waveform driving all 8 combinations with sum and cout tracking a + b + cin.

One clock, computed toggles



Toggle logic feeds a flip-flop bank: D0 = 1, D1 = Q0, D2 = Q1 AND Q0 - trace 000, 001, 010, 011.

HDL and verification: describe, then prove by loop

1. **Module** full_adder: assign sum = a^b^cin; assign cout = (a&b)|(cin&(a^b)).
2. **Testbench** Loop i = 0..7: drive {a,b,cin} = i, assert 2*cout+sum == a+b+cin.
3. **Boundary** One driver per signal; an unchanged path in always @(posedge) infers **memory** - by design.

Check yourself

- K-map: minterms 1,3,5,7 of A,B,C simplify to? (C.)
- Counter: 4-bit synchronous up-counter - when does bit 2 toggle? ($Q1 \cdot Q0 = 11$.)
- HDL: which construct is testbench-only? (#10; the delay.)