Terminology

- Combinational Circuit: a digital circuit whose output value depends solely on the present combination of the circuit inputs' values.
- Sequential Circuit: a digital circuit whose output depends not only on the circuit's present inputs, but also on the circuit's present state.
- Set: input that forces the flip-flop to 1.
- Reset: input that forces the flip-flop to 0.
- Synchronous Inputs: inputs that are used during a rising clock edge.
- Asynchronous Inputs: inputs that are used independent of the clock signal.
- Race Condition: a situation in which the final output of a sequential circuit depends on the delays of gates and wires.
- Glitch: a temporary unintended signal value caused by circuit delays.
- Setup Time: inputs of a flip-flop must be stable for a minimum amount of time before a clock edge arrives.
- Hold Time: the inputs of a flip-flop must remain stable for a minimum amount of time after a clock edge arrives.

Reasons for Gate and Circuit Component Delays

- Transistors don't switch from nonconducting to conducting (or vice versa) immediately.
- Electric current travels at the speed of light.
- Wires can slow down electric current because of characteristics like capacitance and inductance.

Metastability & Output Glitches

- A metastable state is in a state other than a stable 0 or a stable 1.
- When the clock period is sufficiently longer than the longest path, we can ensure the circuit obeys setup and hold times.
- Glitching is the presence of temporary values on a wire typically caused by different delays of different logic paths leading to that wire.
- An output with a flip-flop added is called a registered output.
- A drawback of registered outputs, however, is that the outputs use extra flip-flops in additional to being shifted by one clock cycle.

Decimal-to-Binary (Addition Method)

$(65)_{10}$

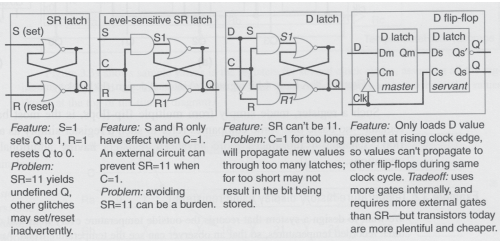
1	0	0	0	0	0	1
64	32	16	8	4	2	1

$64 + 1 = 65 = (1000001)_2$

Clock Frequency

$$f_{clk} = \frac{1}{T}$$

Bit Storage Blocks



Feature: S=1 sets Q to 1, R=1 resets Q to 0. *Problem:* SR=11 yields undefined Q, other glitches may set/reset inadvertently.

Feature: S and R only have effect when C=1. An external circuit can prevent SR=11 when C=1. *Problem:* avoiding SR=11 can be a burden.

Feature: SR can't be 11. *Problem:* C=1 for too long will propagate new values through too many latches; for too short may not result in the bit being stored.

Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle. *Tradeoff:* uses more gates internally, and requires more external gates than SR—but transistors today are more plentiful and cheaper.

Simple Register (4-bit)

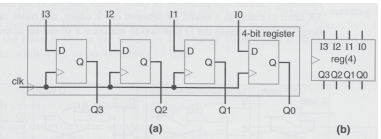


Figure 3.36 A basic 4-bit register: (a) internal design, (b) block symbol.

Parallel-Load Register (4-bit)

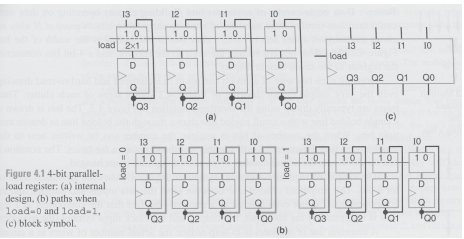


Figure 4.1 4-bit parallel-load register: (a) internal design, (b) paths when `load=0` and `load=1`, (c) block symbol.

Shift Register (4-bit)

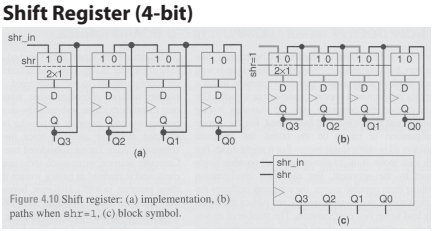


Figure 4.10 Shift register: (a) implementation, (b) paths when `shr=1`, (c) block symbol.

Multifunction Register (4-bit)

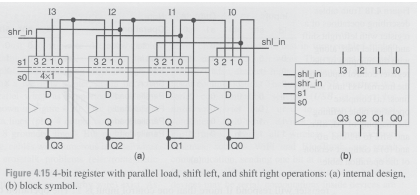


Figure 4.15 4-bit register with parallel load, shift left, and shift right operations: (a) internal design, (b) block symbol.

Decoder (2x4)

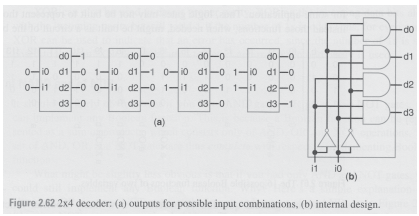


Figure 2.62 2x4 decoder: (a) outputs for possible input combinations, (b) internal design.

Multiplexer (4x1)

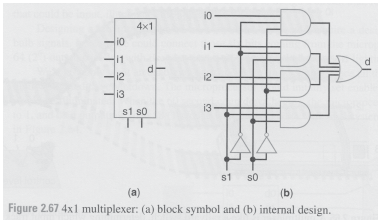


Figure 2.67 4x1 multiplexer: (a) block symbol and (b) internal design.

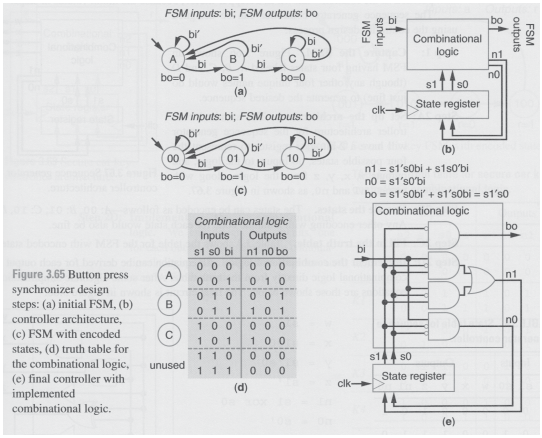


Figure 3.65 Button press synchronizer design steps: (a) initial FSM, (b) controller architecture, (c) FSM with encoded states, (d) truth table for the combinational logic, (e) final controller with implemented combinational logic.

Decimal-to-Two's Complement

1. Convert decimal to binary.
2. Pad to requested bit length (if needed).
3. Invert the bits.
4. Add 1.

NOTE: Follow "all" these steps ONLY IF given a negative decimal number. If positive, simply convert to binary.

Two's Complement-to-Decimal

1. Invert the bits.
2. Add 1.
3. Convert binary to decimal.
4. Add negative sign.

NOTE: Follow "all" these steps ONLY IF a 1 appears is the MSB position. If 0, simply convert to decimal.

Overflow Rules

1. If the sum of two positive numbers yields a negative result, the sum has overflowed.
2. If the sum of two negative numbers yields a positive result, the sum has overflowed.
3. Otherwise, the sum has not overflowed.

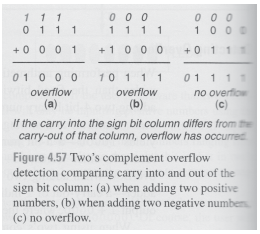


Figure 4.57 Two's complement overflow detection comparing carry into and out of the sign bit column: (a) when adding two positive numbers, (b) when adding two negative numbers, (c) no overflow.

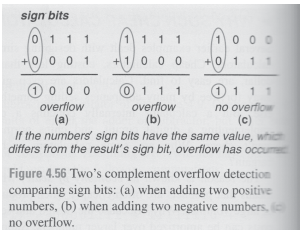


Figure 4.56 Two's complement overflow detection comparing sign bits: (a) when adding two positive numbers, (b) when adding two negative numbers, (c) no overflow.

Facts

- Latches have shorter delays than flip-flops.

Half Adder

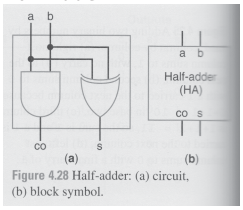


Figure 4.28 Half-adder: (a) circuit, (b) block symbol.

Full Adder

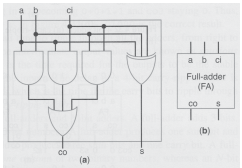


Figure 4.30 Full-adder: (a) circuit, (b) block symbol.

Carry-Ripple Adder

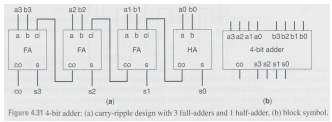


Figure 4.31 4-bit adder: (a) carry-ripple design with 3 full-adders and 1 half-adder, (b) block symbol.

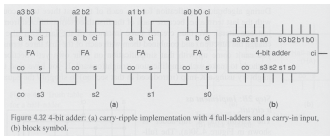


Figure 4.32 4-bit adder: (a) carry-ripple implementation with 4 full-adders and a carry-in input, (b) block symbol.

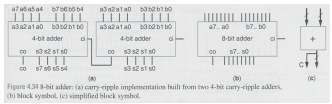


Figure 4.34 8-bit adder: (a) carry-ripple implementation built from two 4-bit carry-ripple adders, (b) block symbol, (c) simplified block symbol.

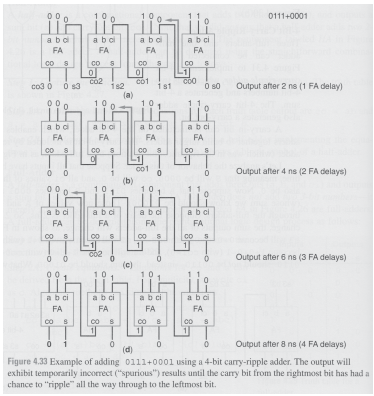


Figure 4.33 Example of adding `0111+0001` using a 4-bit carry-ripple adder. The output will exhibit temporarily incorrect ("spurious") results until the carry bit from the rightmost bit has had a chance to "ripple" all the way through to the leftmost bit.

Incrementor

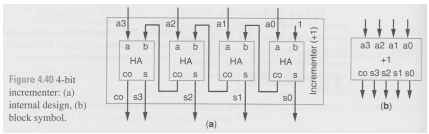


Figure 4.40 4-bit incrementer: (a) internal design, (b) block symbol.

Equality Comparator

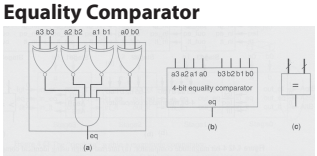


Figure 4.41 Equality comparator: (a) internal design, (b) block symbol, (c) simplified symbol.

Multiplier

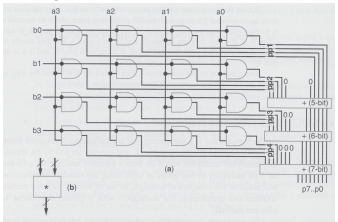


Figure 4.45 4-bit by 4-bit array-style multiplier: (a) internal design, (b) simplified block symbol.

```
                a3    a2    a1    a0
              x b3    b2    b1    b0
-------------------------------------
                   b0a3  b0a2  b0a1  b0a0    (pp1)
+            b1a3  b1a2  b1a1  b1a0    0     (pp2)
+      b2a3  b2a2  b2a1  b2a0    0     0     (pp3)
+ b3a3  b3a2  b3a1  b3a0    0     0     0     (pp4)
-------------------------------------
p7  p6    p5    p4    p3    p2    p1    p0
```

Subtractor

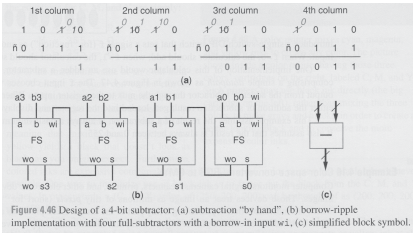


Figure 4.46 Design of a 4-bit subtractor: (a) subtraction "by hand", (b) borrow-ripple implementation with four full-subtractors with a borrow-in input `wi`, (c) simplified block symbol.

Magnitude Comparator

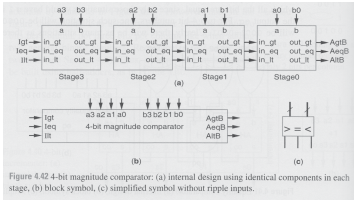


Figure 4.42 4-bit magnitude comparator: (a) internal design using identical components in each stage, (b) block symbol, (c) simplified symbol without ripple inputs.

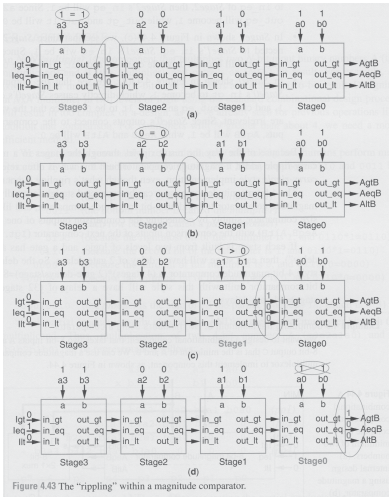


Figure 4.43 The "rippling" within a magnitude comparator.