Algorithmic State Machine (1A)

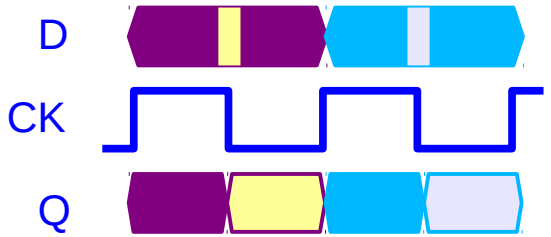
Please send corrections (or suggestions) to youngwlim@hotmail.com.

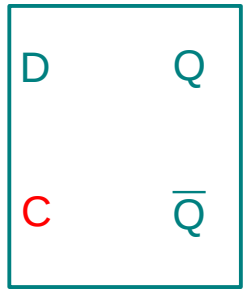
This document was produced by using LibreOffice and Octave.

D Latch & D FlipFlop

Level Sensitive D Latch

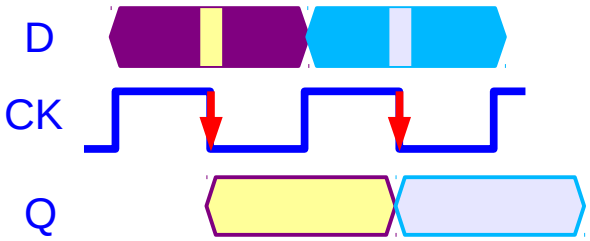
CK=1 transparent
CK=0 opaque

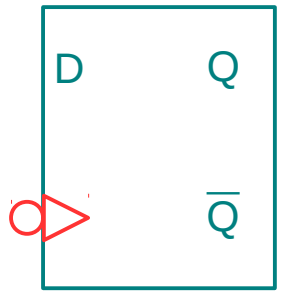




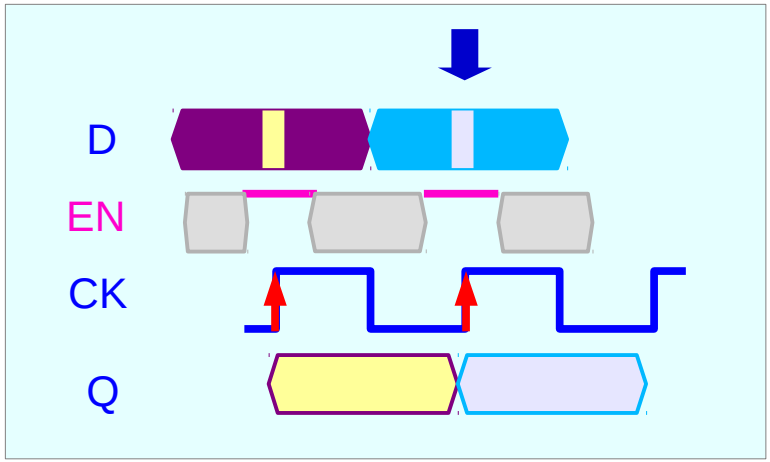
Edge Sensitive D FlipFlop

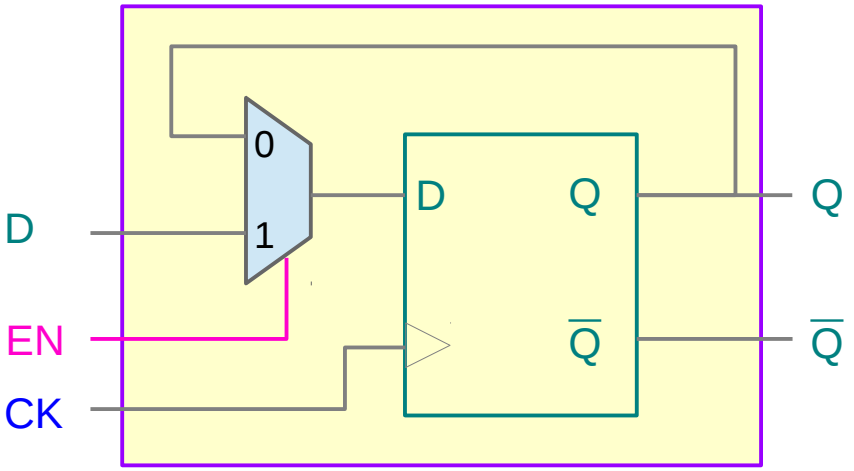
CK=1→0 transparent
else opaque

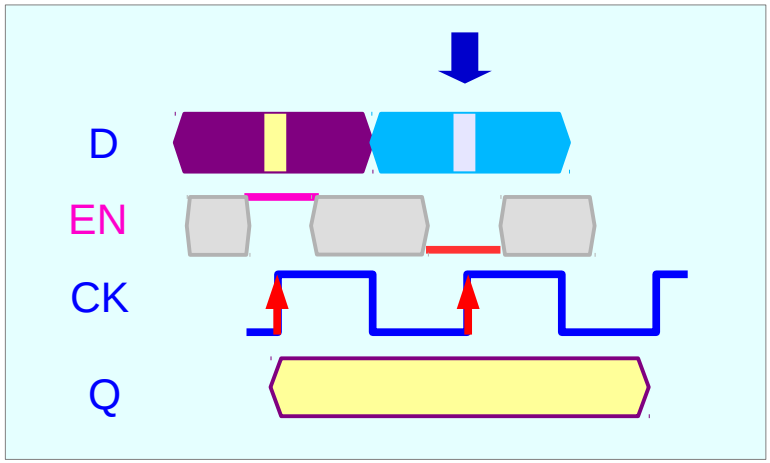


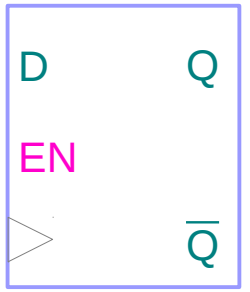


D FlipFlop with Enable

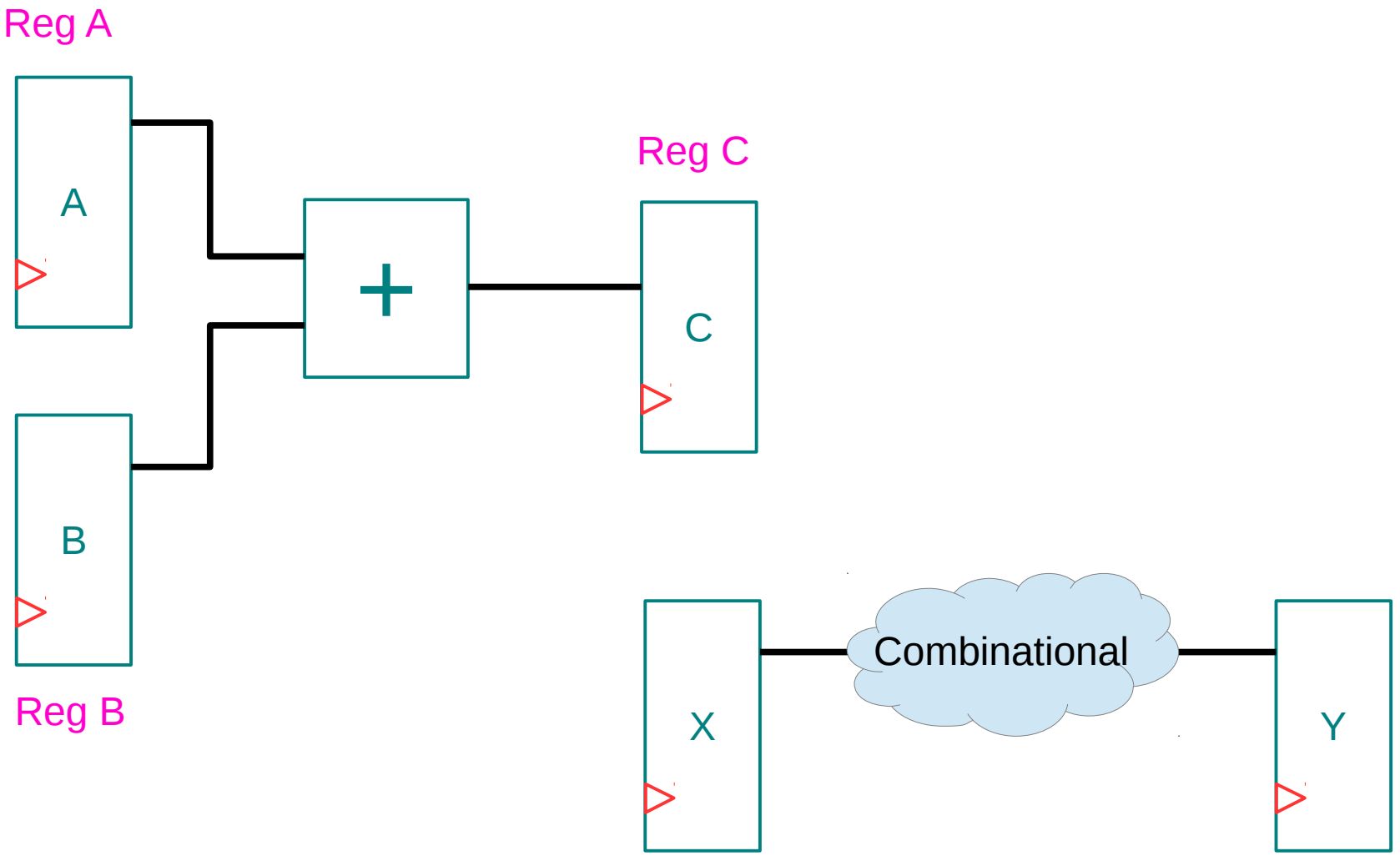




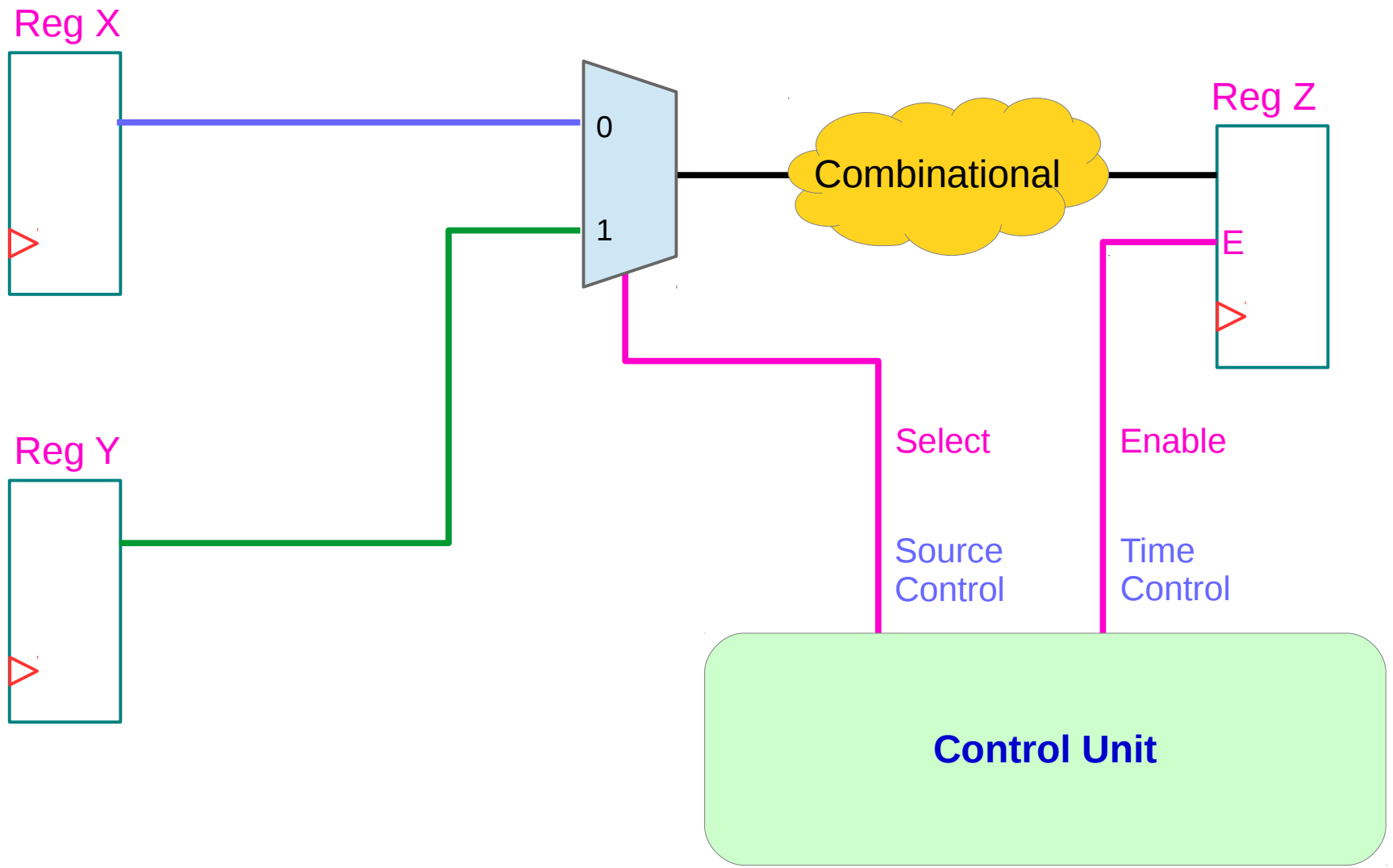




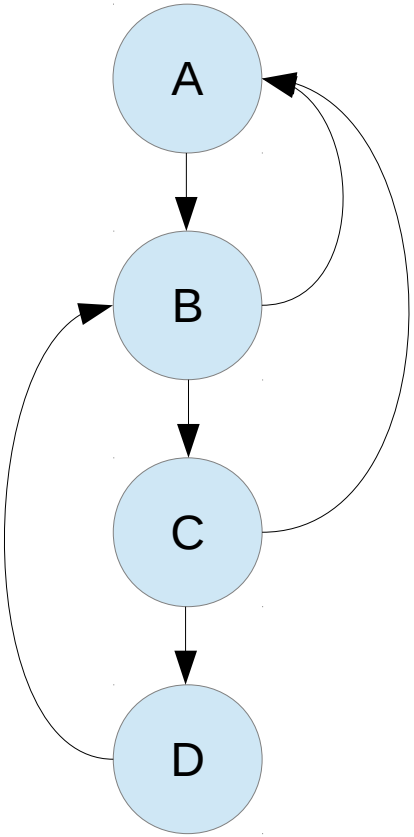
Register Transfer & Data Path

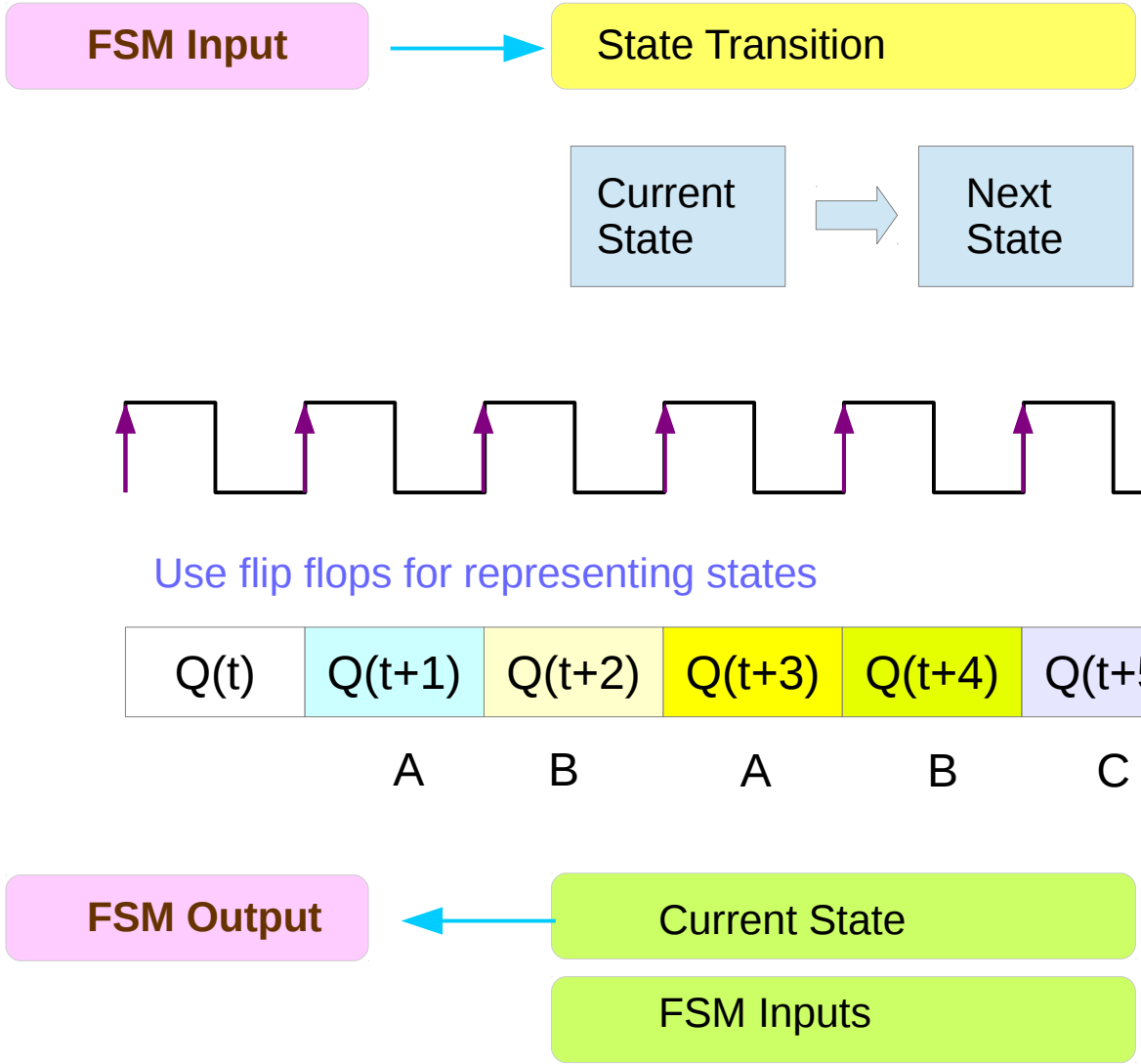


Control the time & the source of a transfer

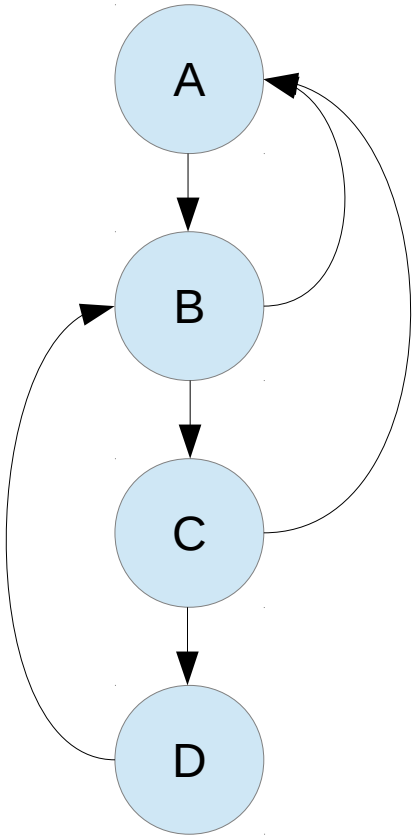


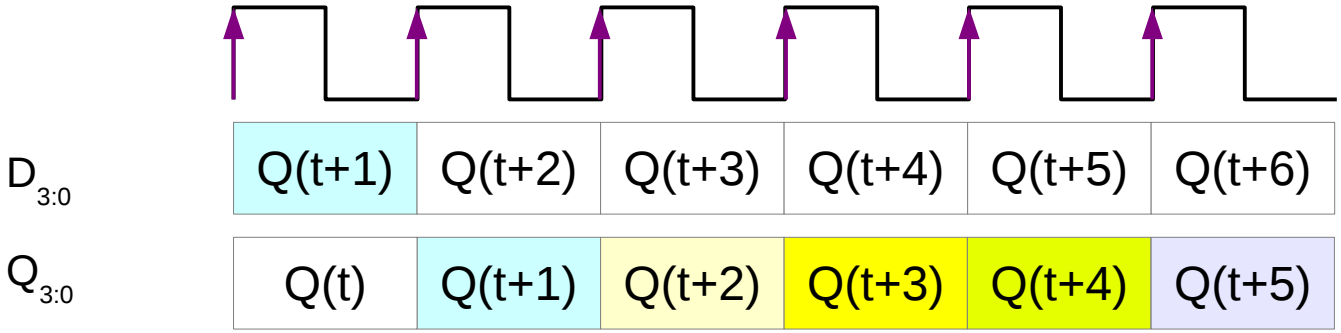
Finite State Machine

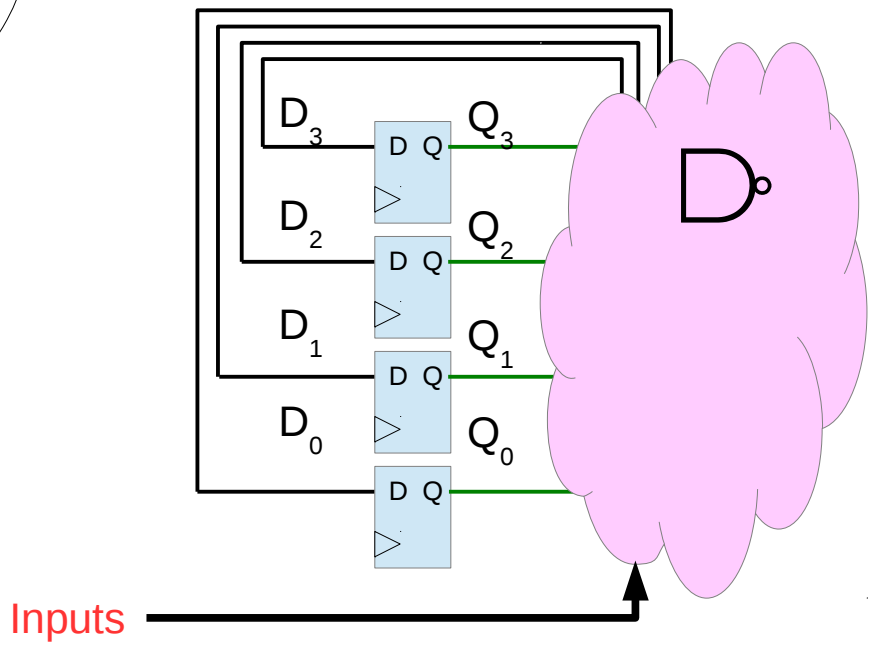


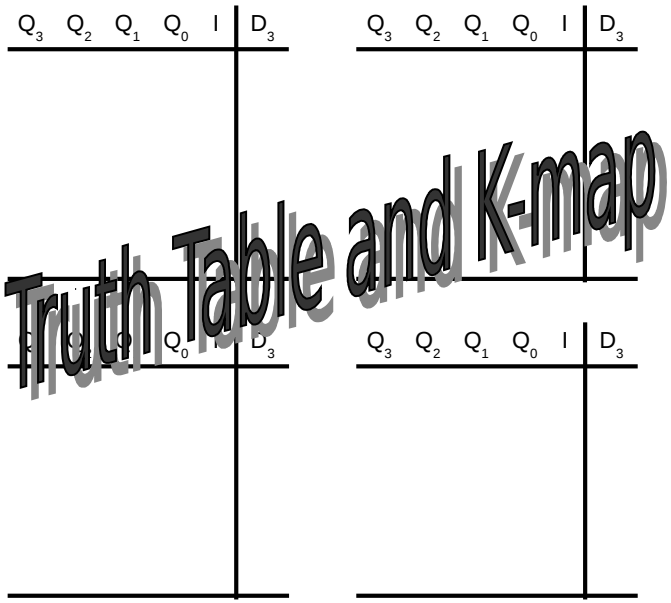


Finding Next State Logic

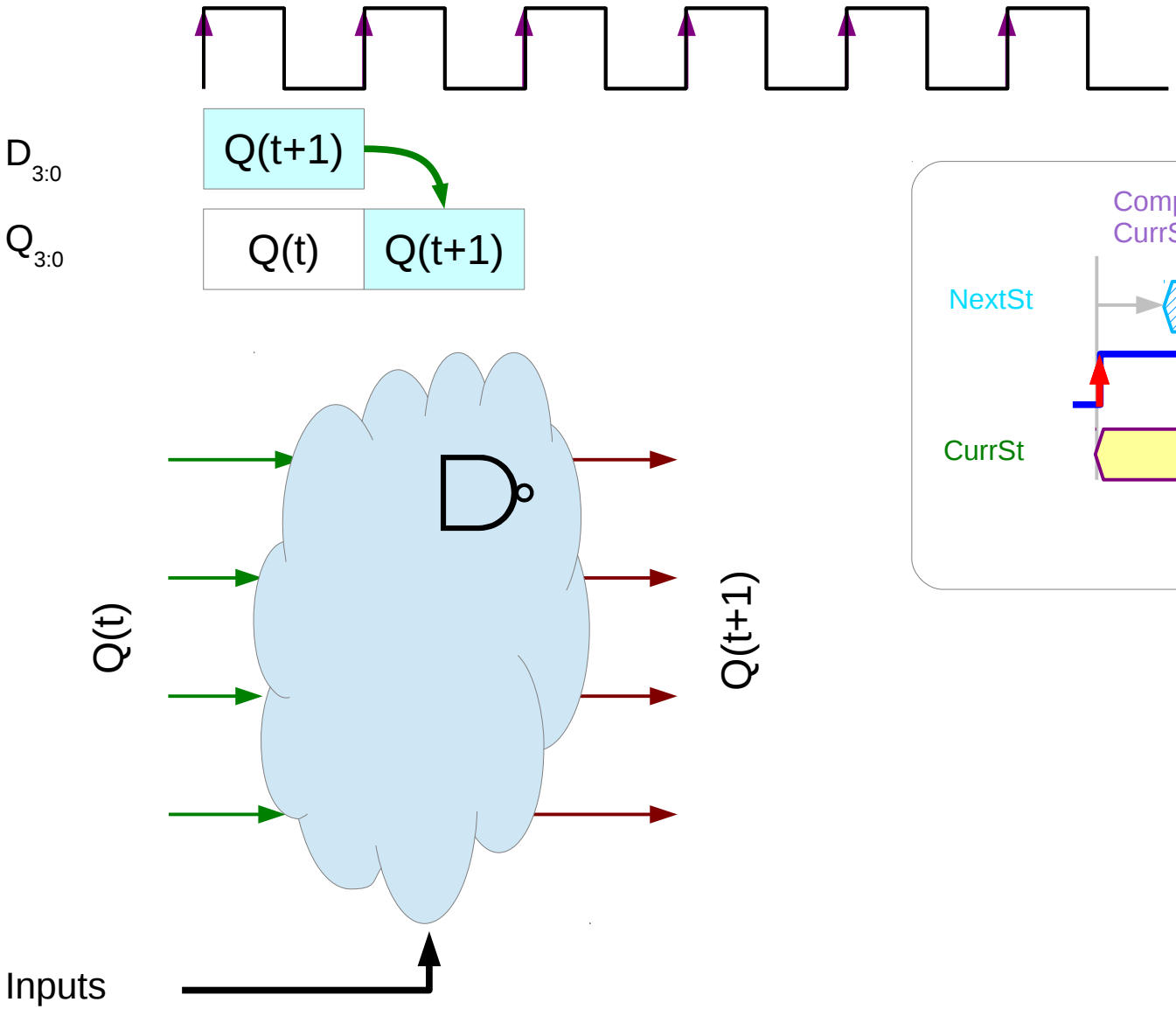


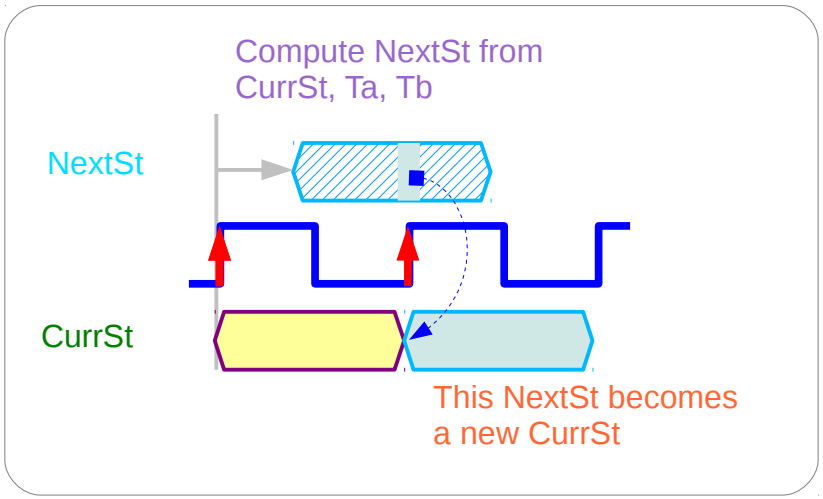


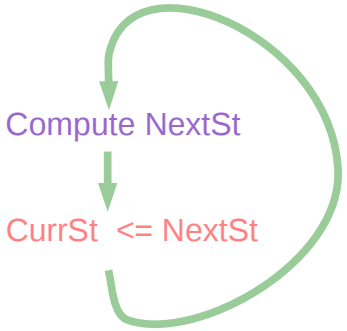




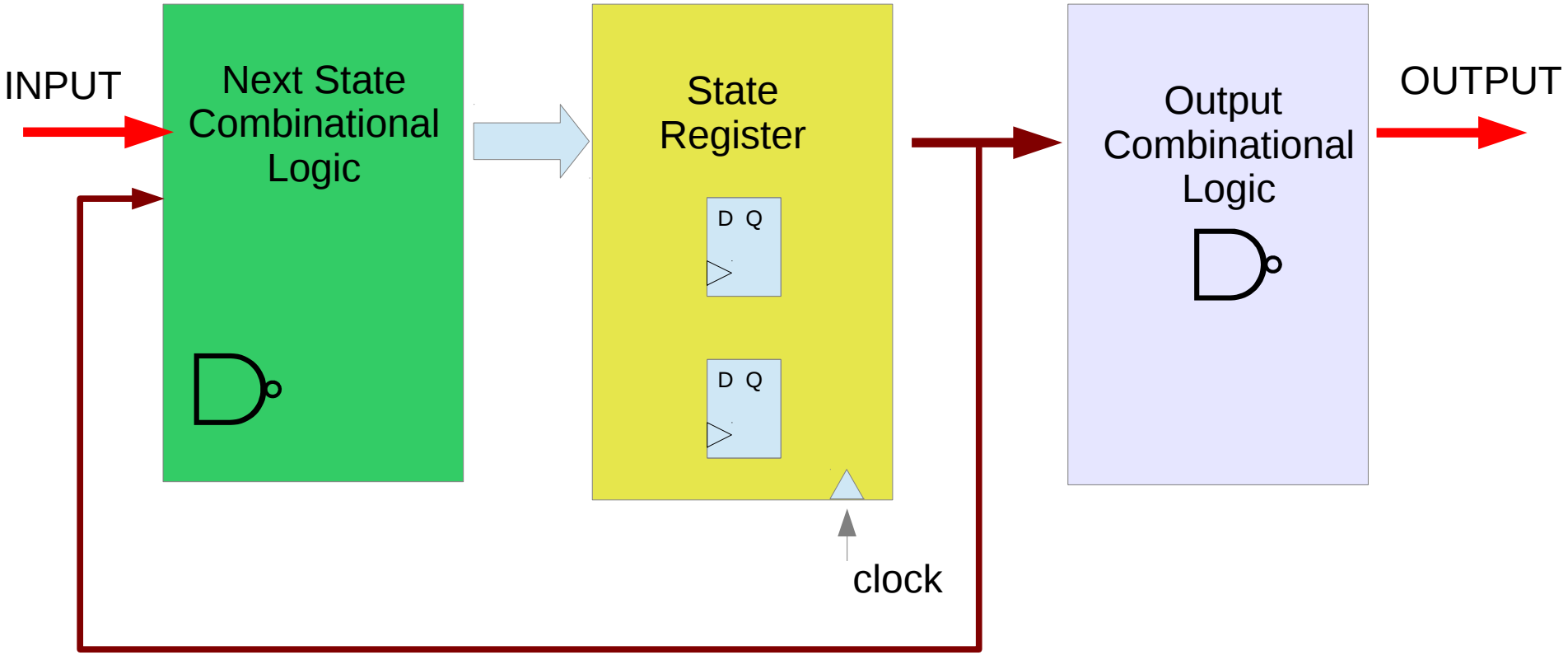
State Transition



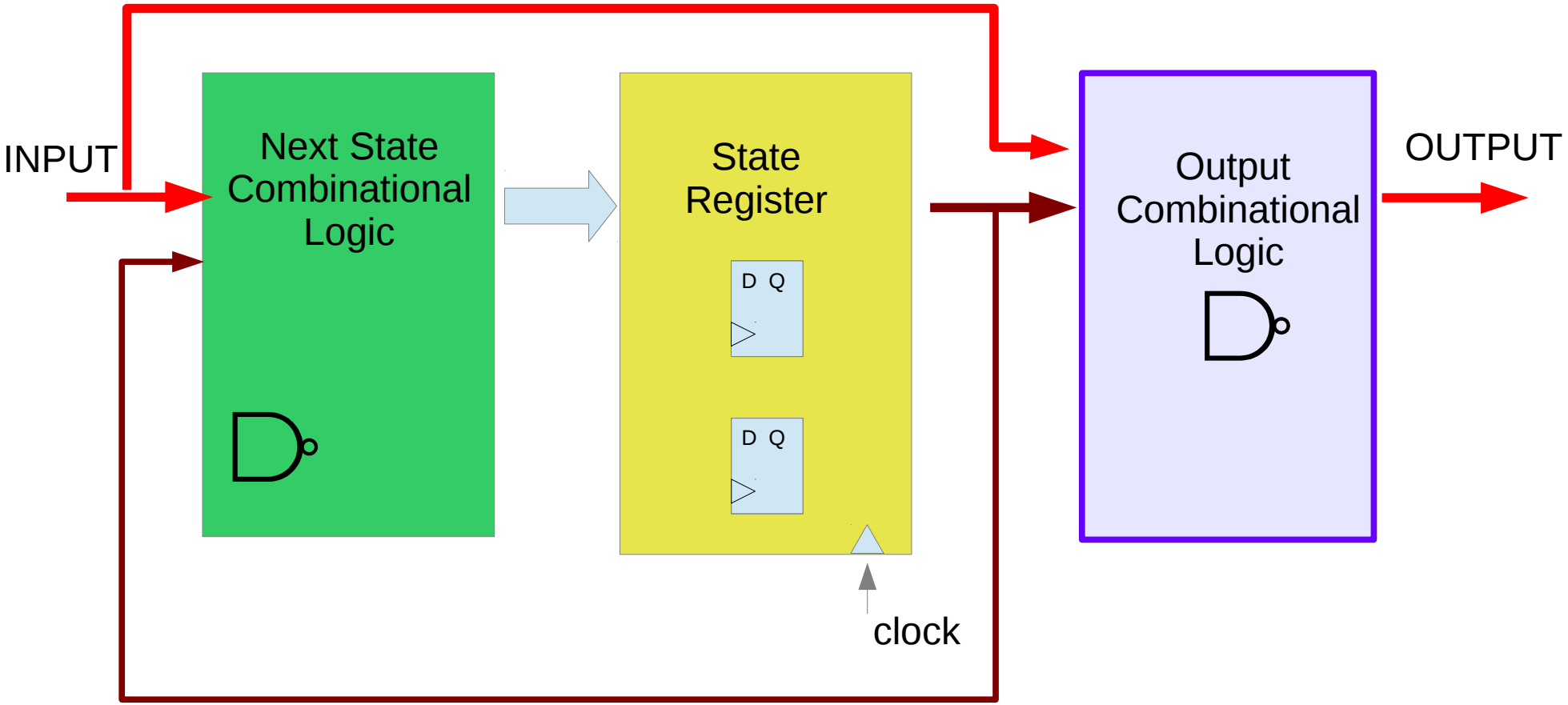




Moore FSM

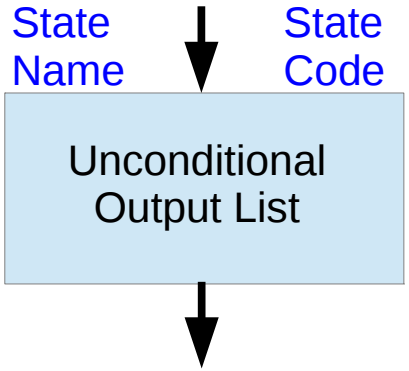

INPUT
Next State
Combinational
Logic
State
Register
D Q
D Q
Output
Combinational
Logic
OUTPUT
clock

Mealy Machine

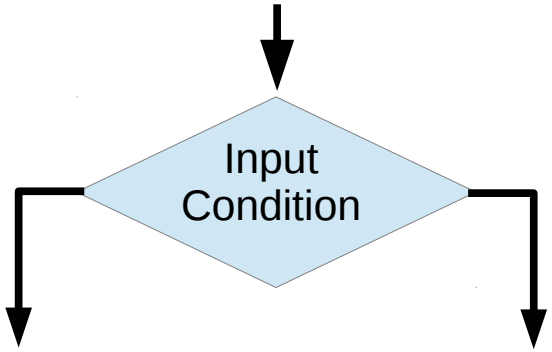


Flow Chart & Algorithmic State Machine

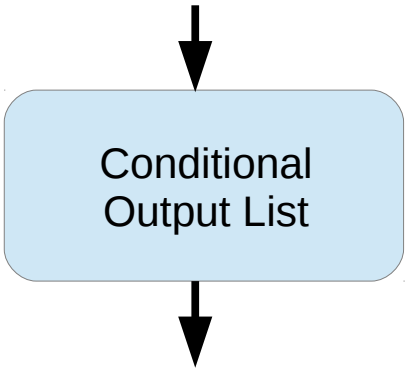
State Box



Condition Symbol



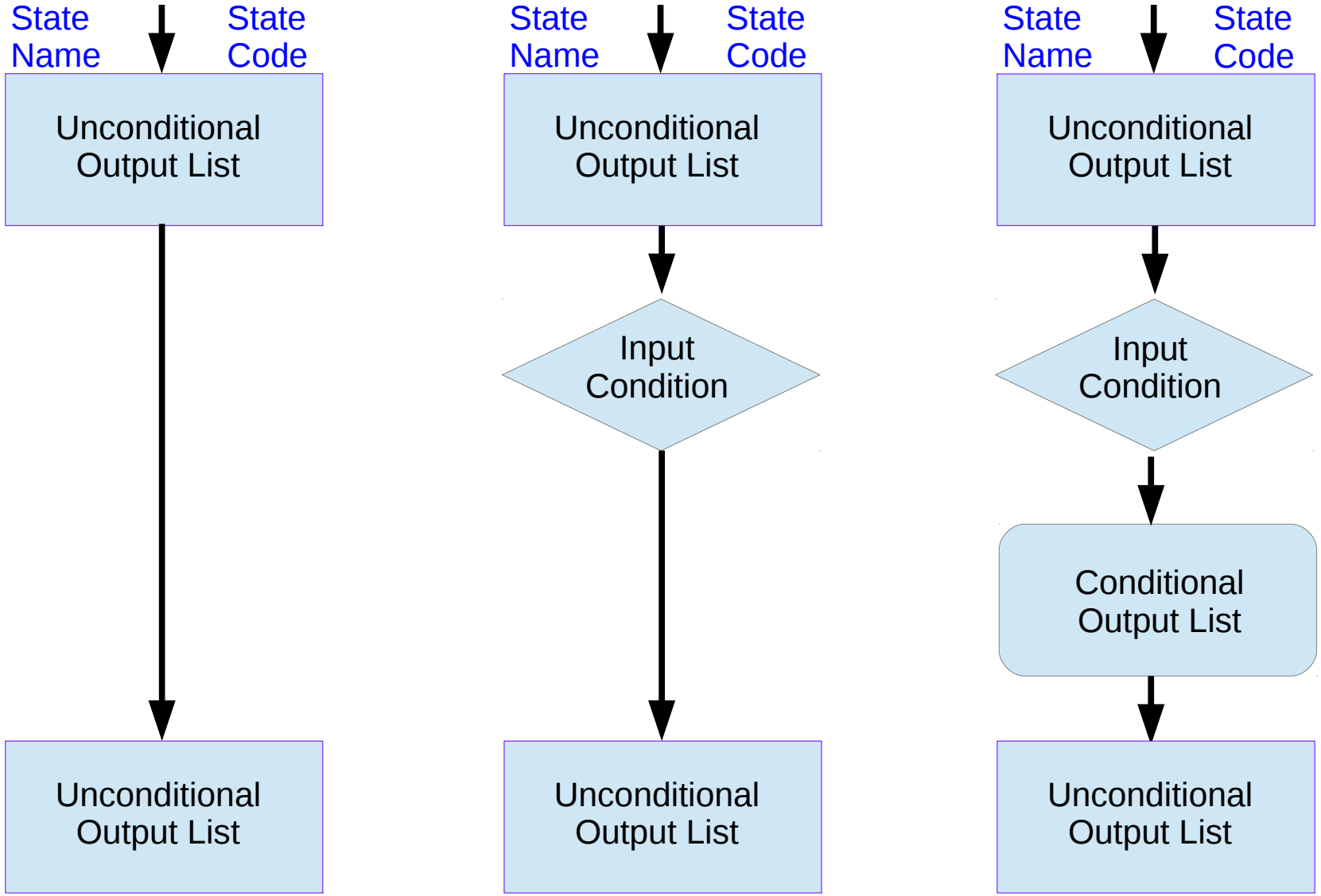
Conditional Output Box



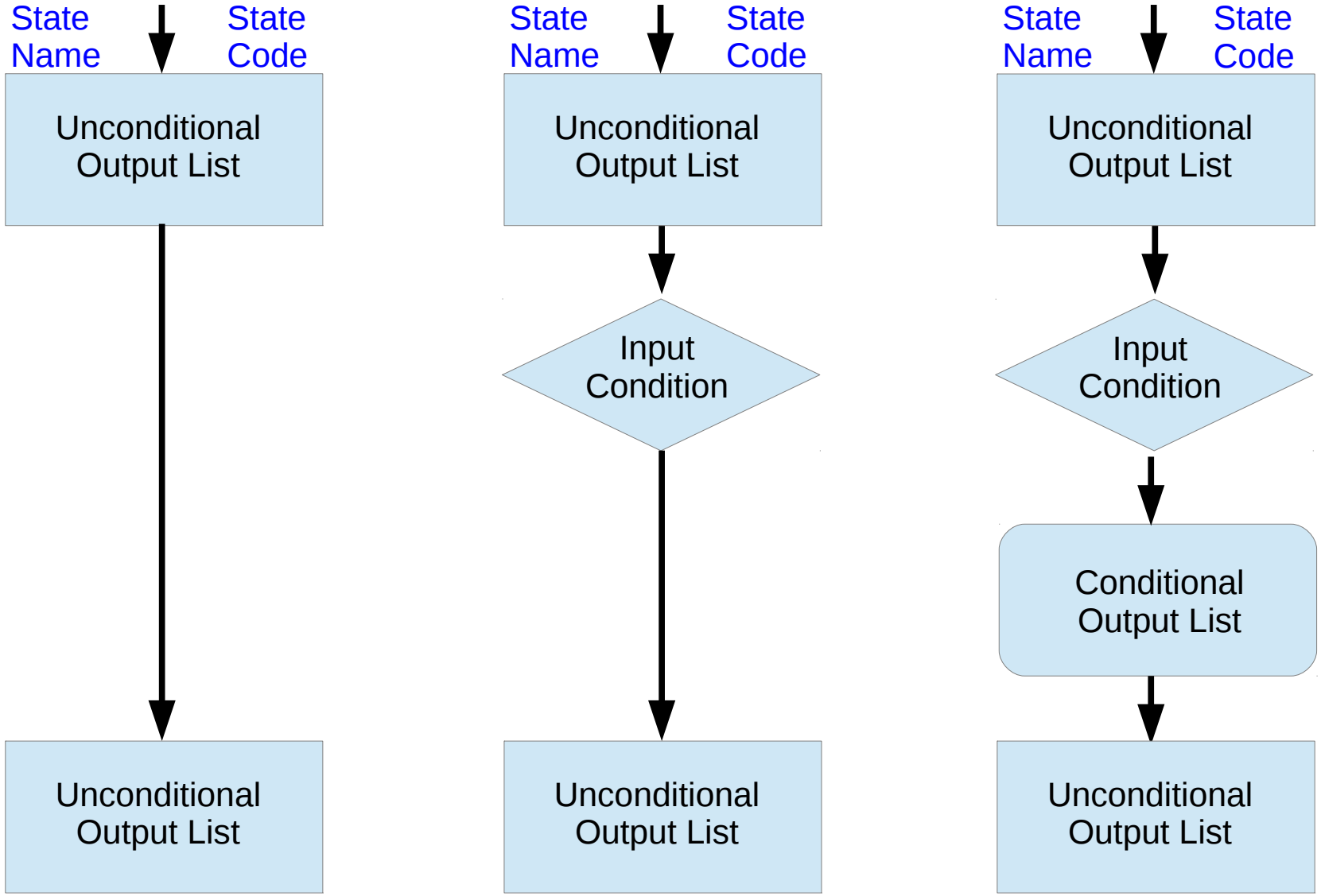
Moore Output

Mealy Output

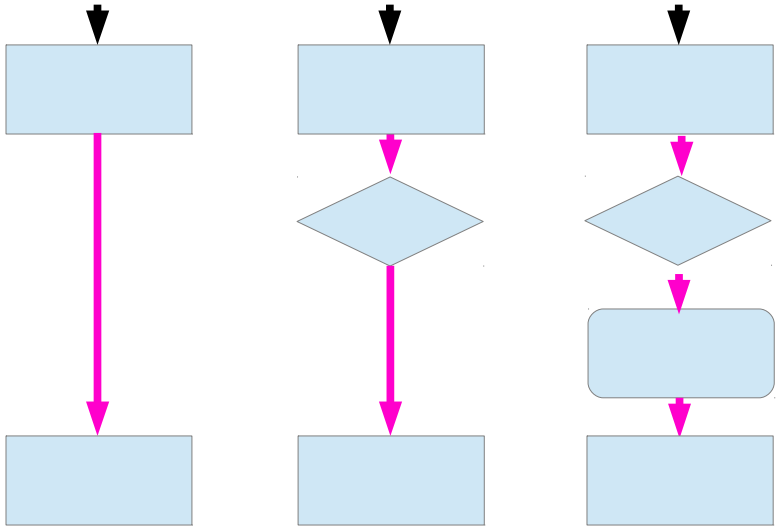
Restriction



Restriction



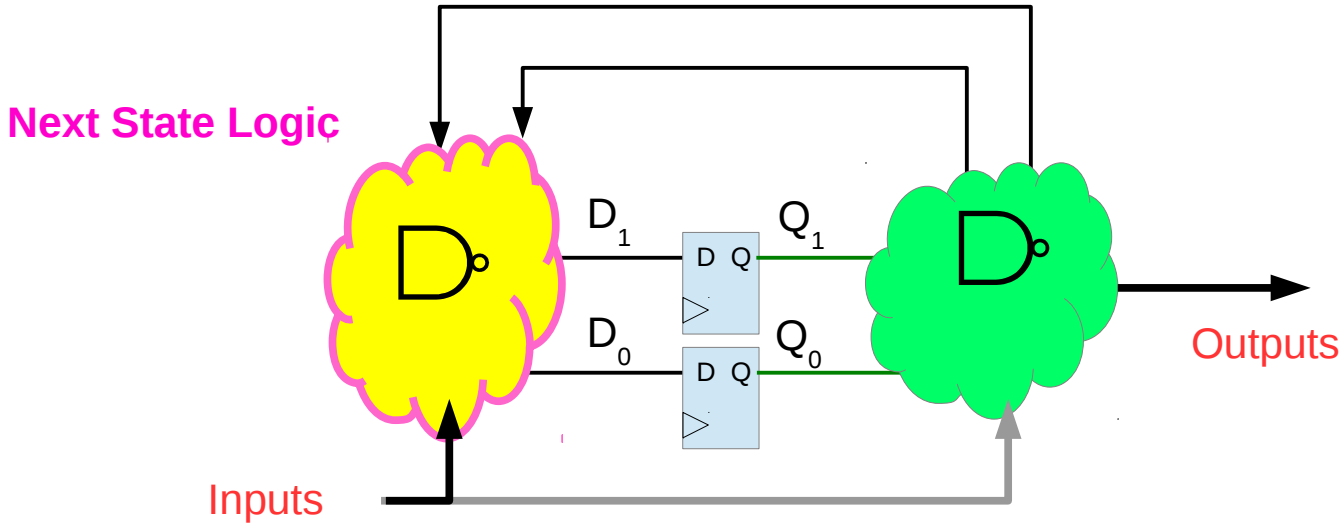
Next State Logic



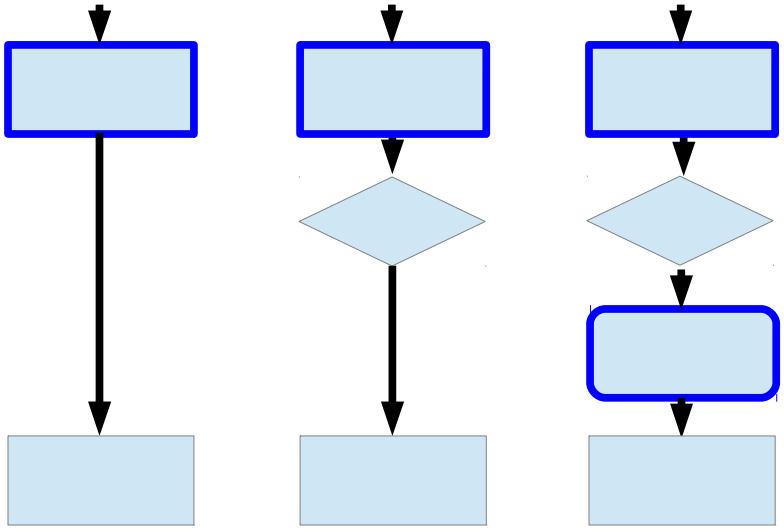
FSM Input

State Transition

Current State → Next State



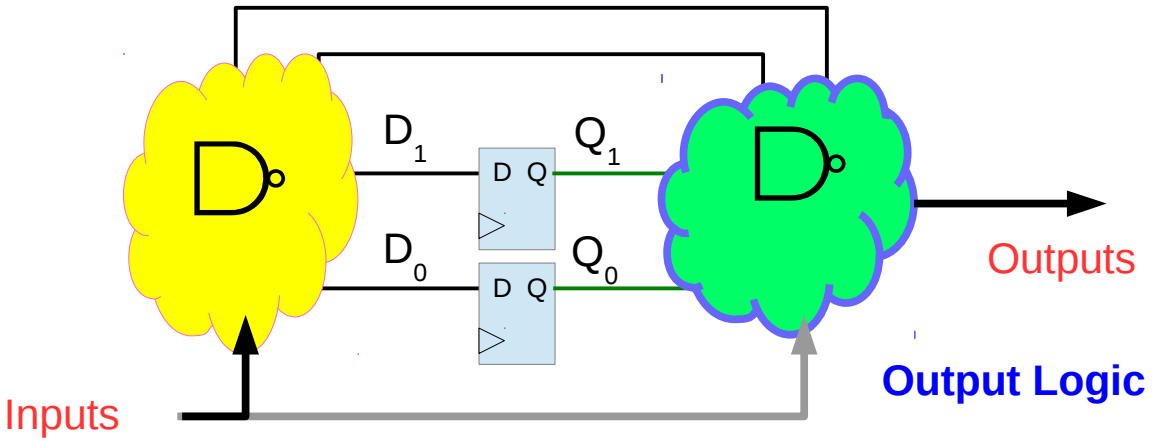
Output Logic



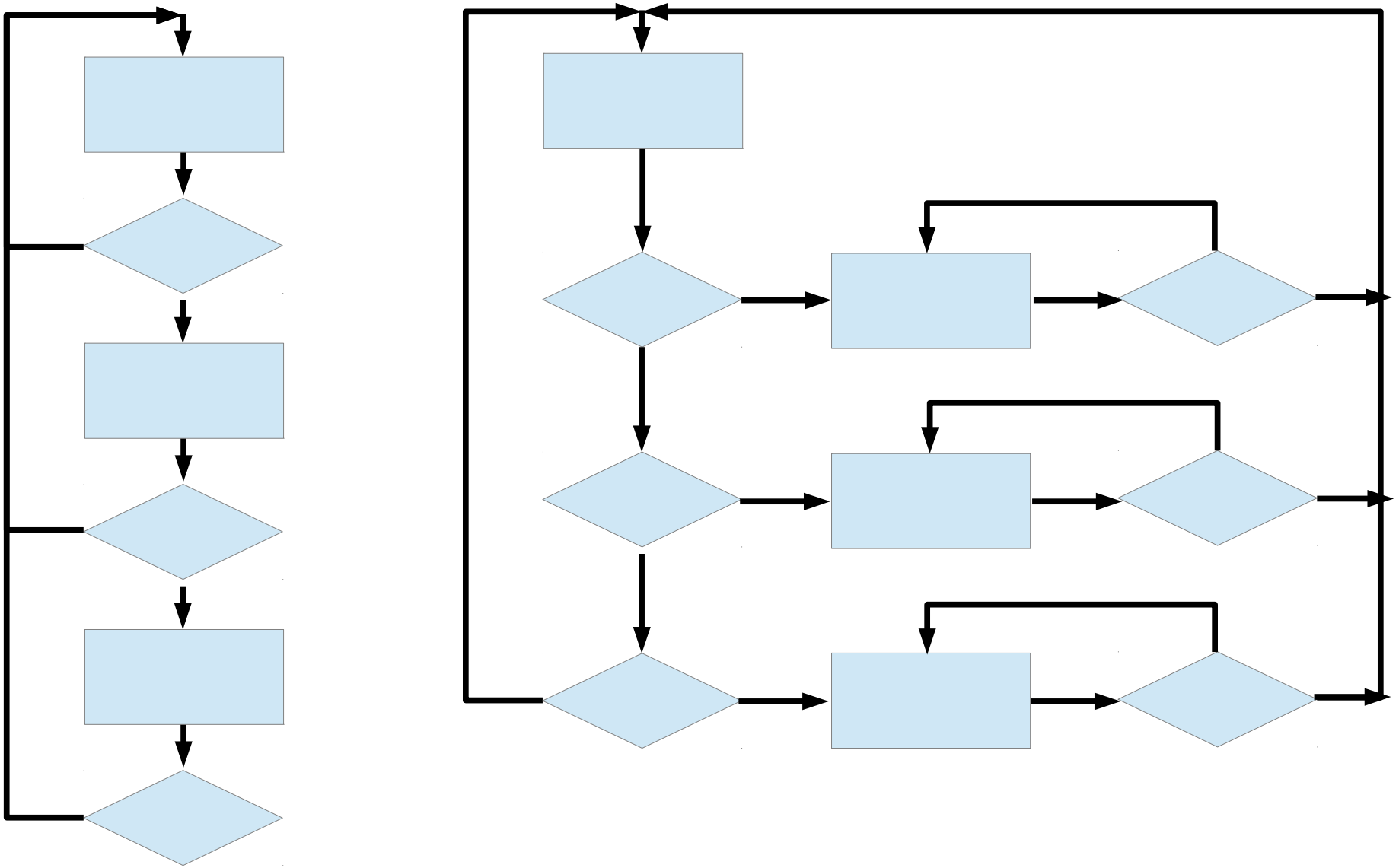
FSM Output

Current State

FSM Inputs



Examples



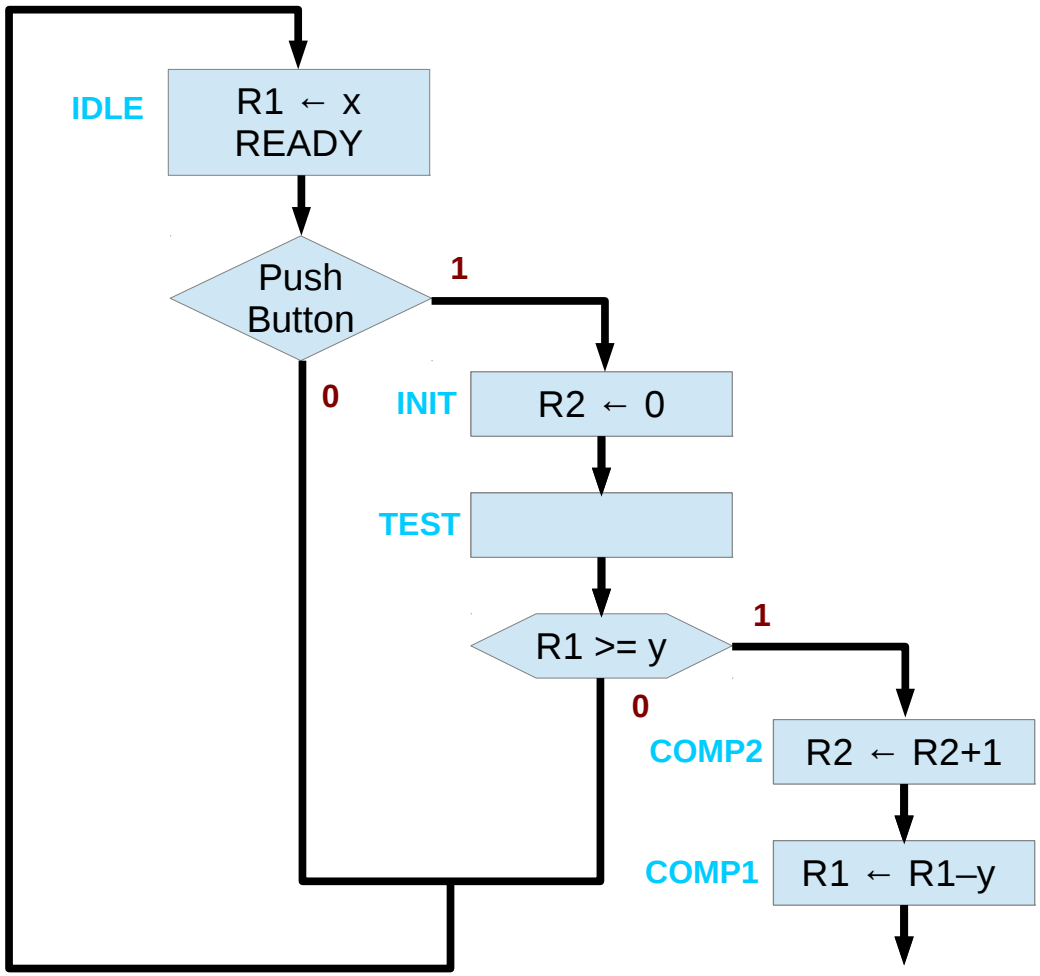
Examples

```
int x = 10, y = 3;

int R1 = x;
int R2 = 0;

while (R1 >= y) {
    R1 = R1 – y ;
    R2 = R2 + 1;
}
```

R1 = x % y
R2 = x / y



References

[1] http://en.wikipedia.org/
[2] M. M. Mano, C. R. Kime, “Logic and Computer Design Fundamentals”, 4th ed.
[3] D.M. Harris, S. L. Harris, “Digital Design and Computer Architecture”
[4] M. G. Arnold, “Verilog Digital Computer Design : Algorithms into Hardware”, 1999
[5] F.P. Prosser, D.E. Winkel, “The Art of Digital Design : An Intro to Top-Down Design”, 2nd ed, 1986