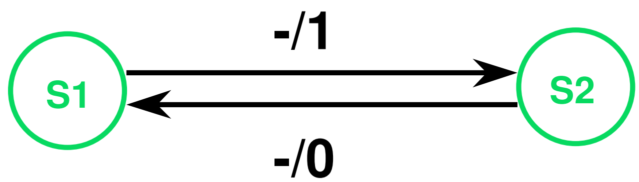
Verifying Sequential Circuits

Sequential circuits can be specified at different levels of abstraction

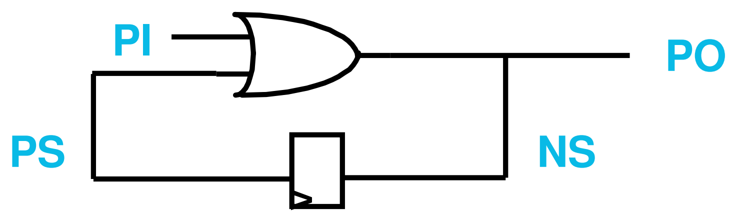
- Register-transfer

```
for (i = 1;  i ≤ 10;  i++)
  a[i]  =  b[i] * 2;
```

- State Transition Graph



- Logic



Sequential Circuit Equivalence

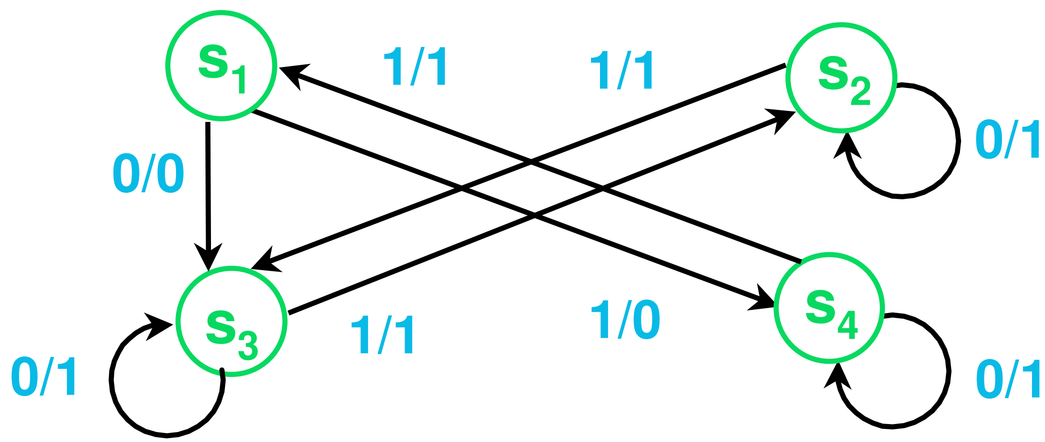
Equivalence between deterministic finite automata representing the two circuits

Two basic approaches

- **Appeal to the canonicity of a reduced, completely specified State Transition Graph.**
- **Equivalence of deterministic finite automata (DFA) by checking to see if they accept the same language.**

State Equivalence

Two states in a State Transition Graph are deemed equivalent if there does not exist an input sequence which differentiates the two states



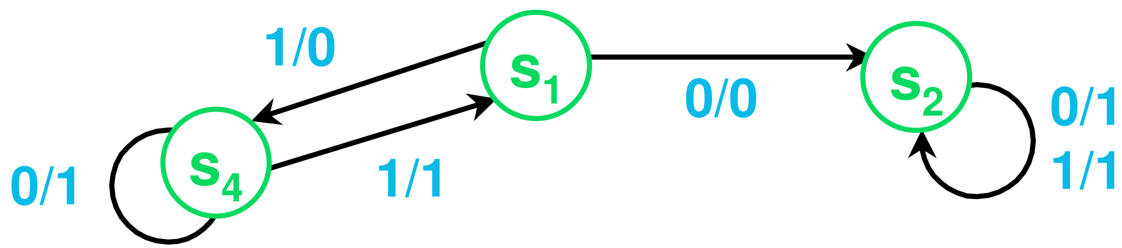
s_1 and s_2 can be differentiated by the vector 0

s_2 and s_3 are equivalent states

Canonicity of a Reduced Machine

Given a completely specified State Transition Graph, a reduced machine is one in which no pair of states is equivalent

Reduced machine is canonical



Can reduce a machine in O(N log(N)) time

Equivalence Checking

Given two STGs G_1 and G_2

If starting states $s_1 \in G_1$ $s_2 \in G_2$ are given:

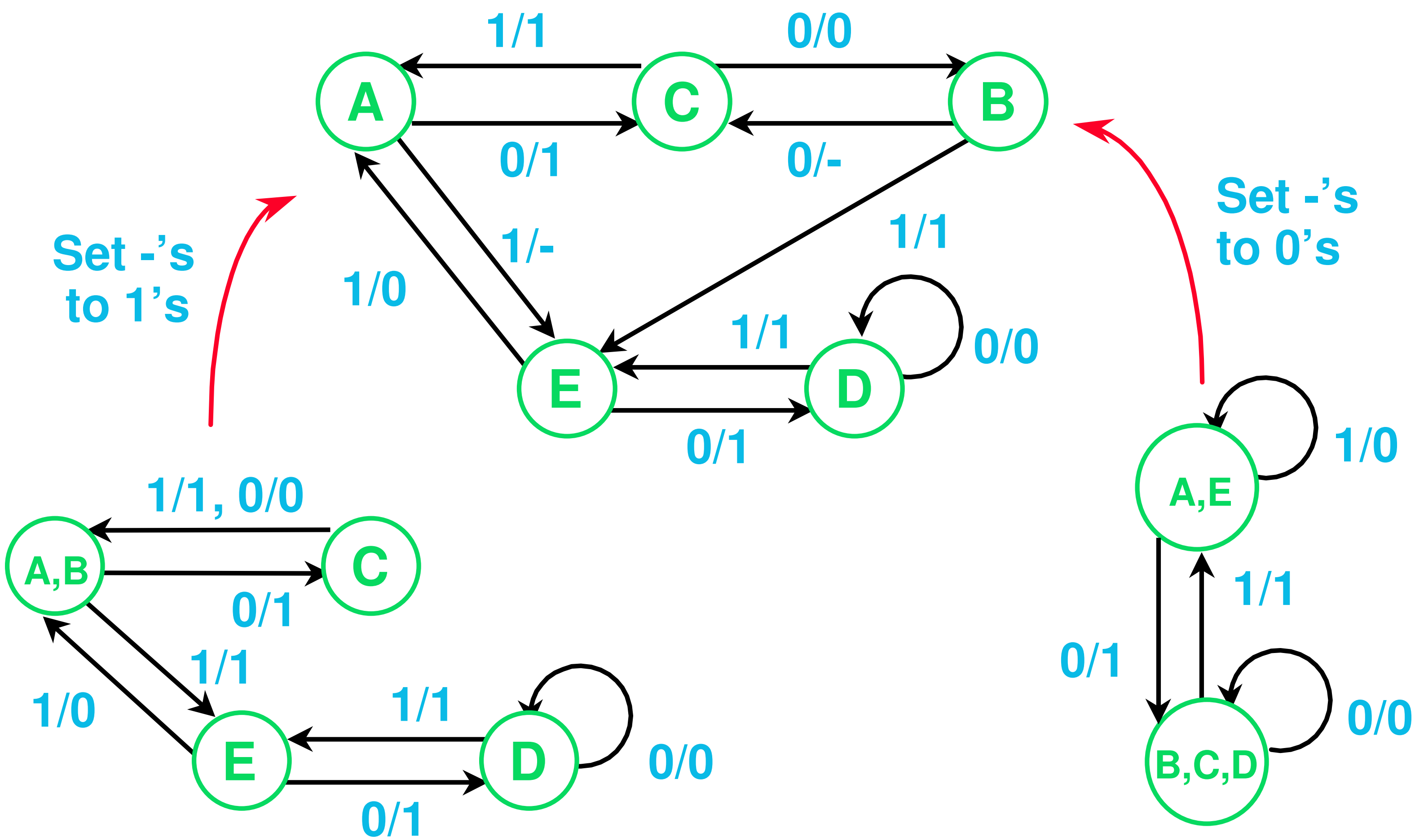
- Reduce G_1 to G_1'
- Reduce G_2 to G_2'
- If $|G_1'| \neq |G_2'|$ then $G_1 \not\equiv G_2$
- Check isomorphism between G_1' and G_2' with (s_1, s_2) as a pair

If starting states are not given:

- Obtain reduced G_1' and G_2'
- If $|G_1'| \neq |G_2'|$ then $G_1 \not\equiv G_2$
- Concatenate G_1', G_2' to get G_3
- Reduce G_3 to get G_3'
- Iff $|G_3'| = |G_1'|$, $G_1 \equiv G_2$

Incompletely-Specified Machines

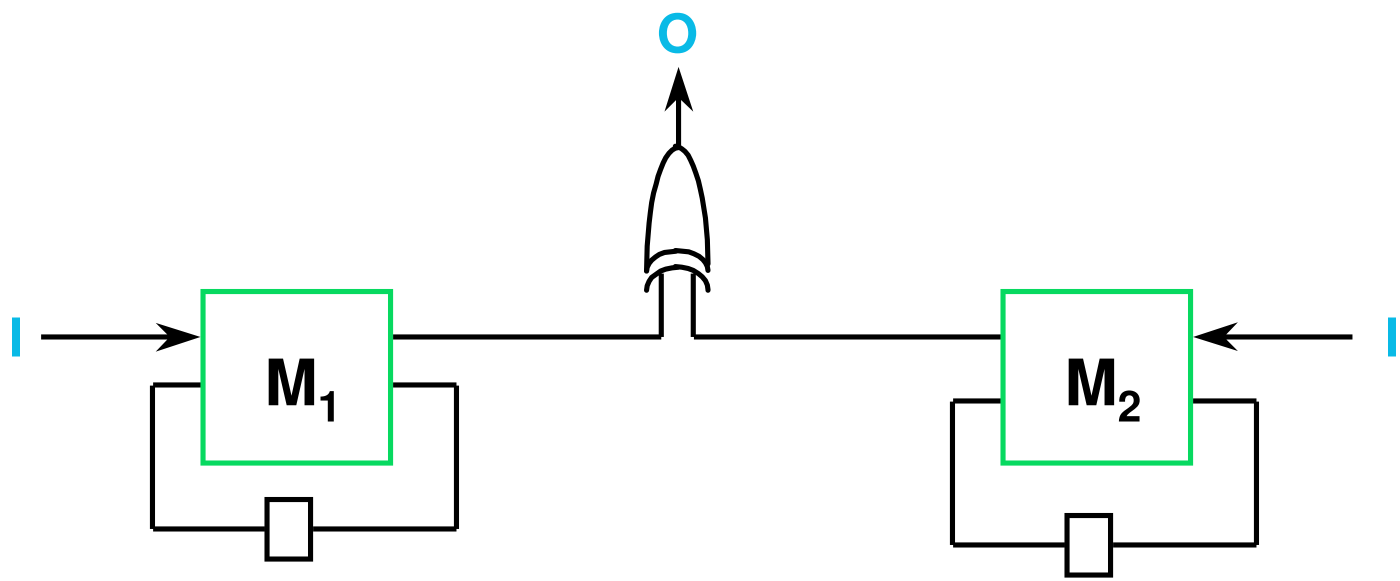
Reduced, incompletely-specified machine is not canonical!



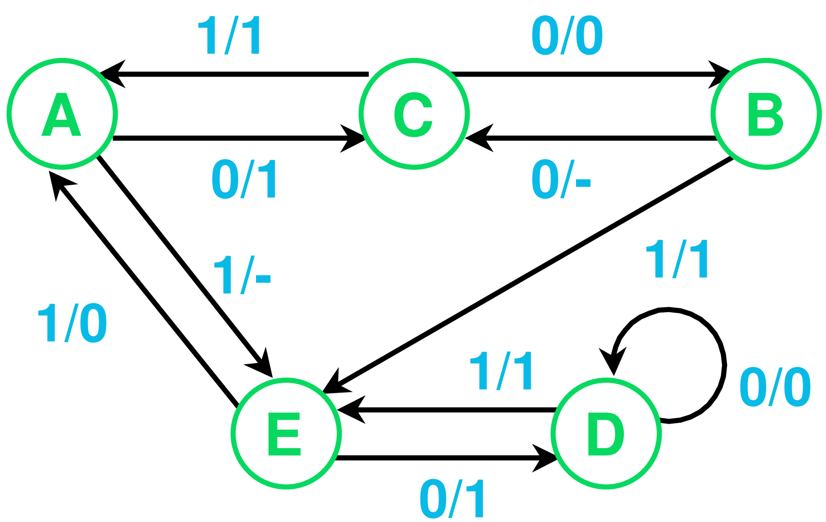
DFA Equivalence

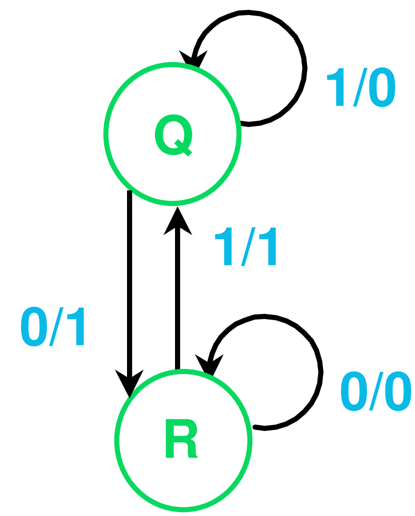
To check equivalence between two (possibly incompletely-specified) STGs, we will construct a product machine.

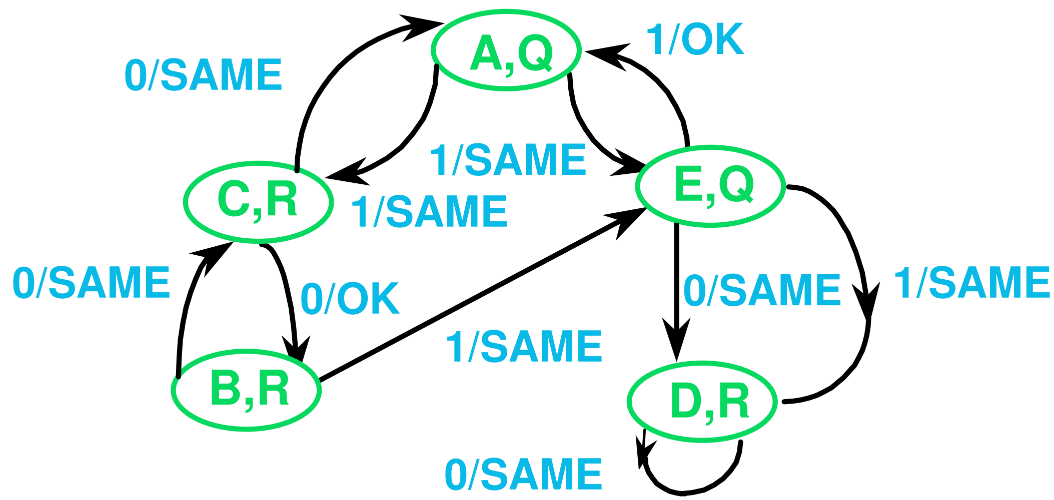
Iff there is a path from the start state of the product machine to a state producing an output of 1, the two machines are not equivalent.



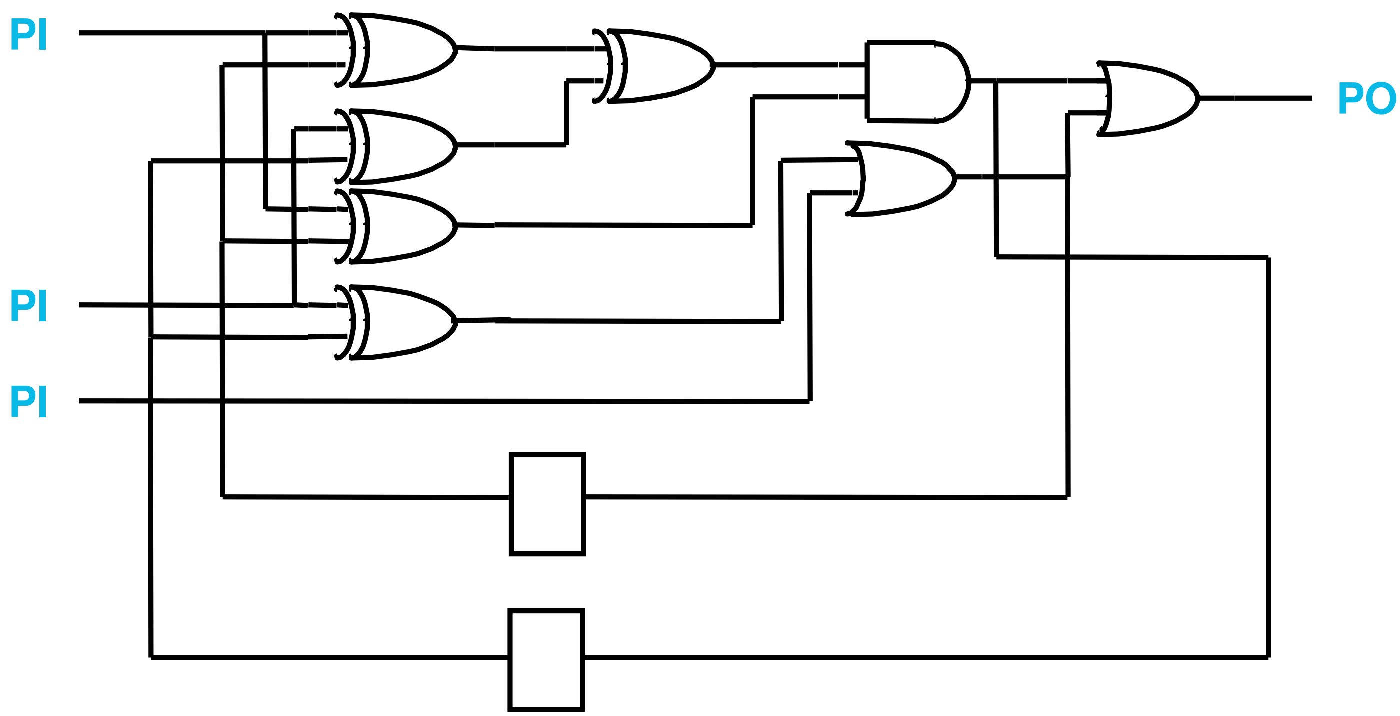
Example


1/1
0/0
A
C
B
0/1
0/-
1/-
1/1
1/0
1/1
0/0
E
D
0/1


Q
1/0
1/1
0/1
0/0
R


0/SAME
A,Q
1/OK
1/SAME
C,R
1/SAME
E,Q
0/SAME
0/OK
0/SAME
1/SAME
1/SAME
B,R
D,R
0/SAME

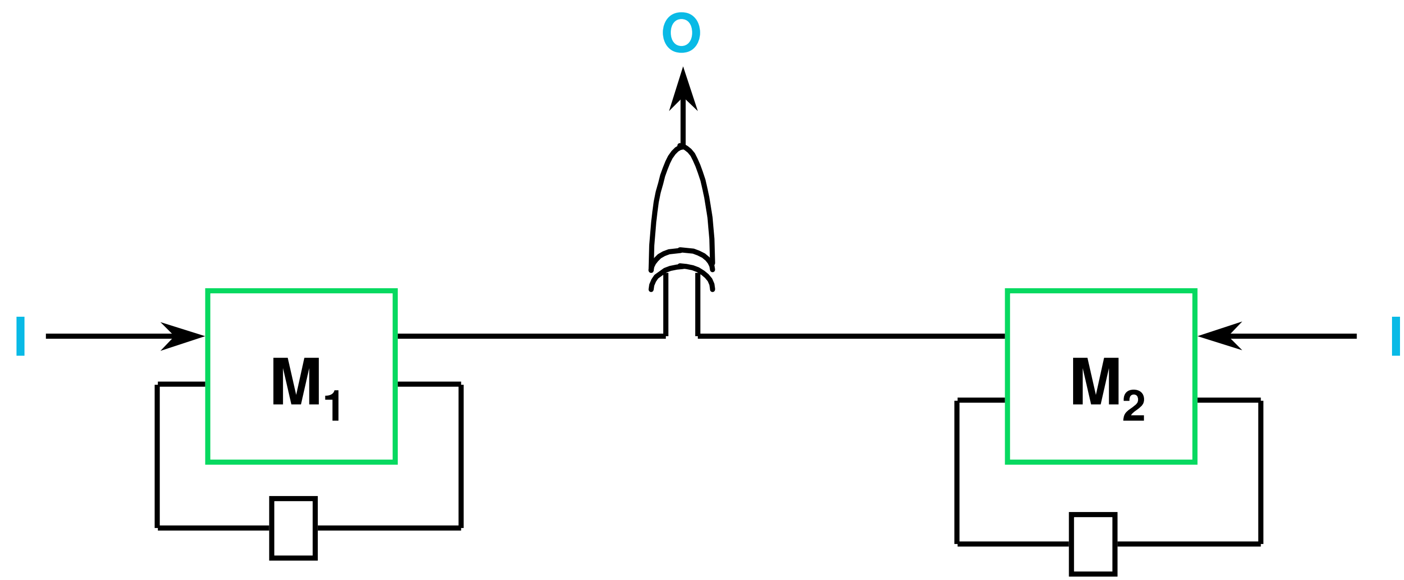
Logic-Level FSMs



Retime or apply precomputation and obtain new logic-level circuit

Could extract STGs, reduce STGs, and check for equivalence

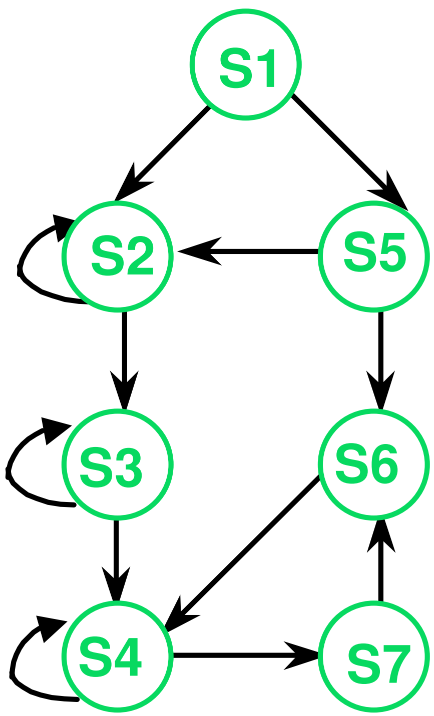
Implicit State Enumeration



Determine reachable states of product machine without explicitly enumerating states

$\Rightarrow$ **Breadth-first enumeration**

Breadth-First Enumeration



Product machine

S1

S1 S2 S5

S1 S2 S5 S3 S6

S1 S2 S5 S3 S6 S4

$\underbrace{\text{S1 S2 S5 S3 S6 S4 S7}}_{\text{entire reachable set}}$

If any state in reachable set has a fanout edge with an output of 1, the machines M_1 and M_2 are not equivalent.

Algorithm

C_0 is the initial state

I is the set of all input combinations

C_R is the set of reachable states

Iteration formula:

$C_{i+1} = C_i \cup f(C_i \times I)$

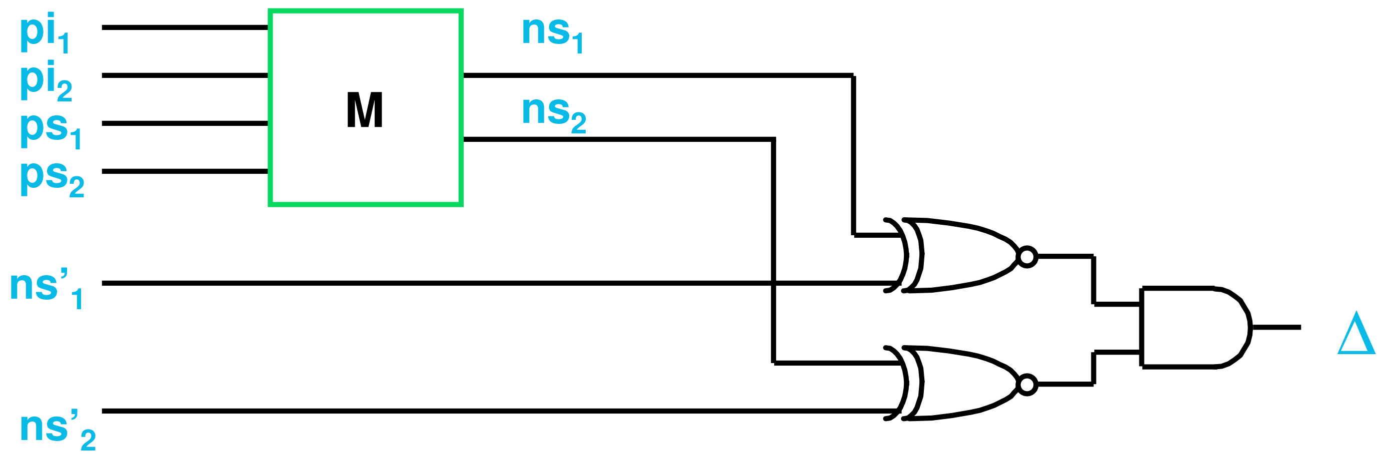
$f(C_i \times I)$ is the set of states reachable from C_i on applying I

Stop when $C_{n+1} = C_n = C_R$

Check if any state in C_R has output 1

Transition Relation

Represent set of states as a single-output Boolean function over the present state lines



$\Delta(PI, PS, NS')$ is the transition relation of M

$\Delta(pi_t, ps_t, ns_t) = 1$ if state ns_t reached on applying pi_t to ps_t is $ns'_t = ns_t$

Smoothed Transition Relation

Consider Δ_s (PS, NS') which is $S_{PI}\ \Delta$(PI, PS, NS')

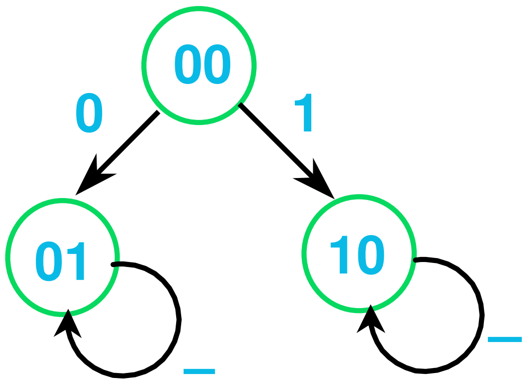
That is, smooth away all primary inputs to M from Δ

$$S_{pij}\,\Delta = \Delta_{pij} + \Delta_{\overline{pij}}$$

$$S_{PI}\,\Delta = S_{pi1}\ S_{pi2}\ \ldots\ S_{piN}\ \Delta$$

Δ_s (PS, NS') will allow us to compute efficiently the <u>set</u> of states reachable from a <u>set</u> of states (without explicit enumeration)

Example Transition Relation



PI	PS	NS'	Δ
0	00	01	1
1	00	10	1
-	01	01	1
-	10	10	1

Δ_s (PS, NS')

PS	NS'	Δ_s
00	01	1
00	10	1
01	01	1
10	10	1

Computing the Reachable Set

Given Δ_s (PS, NS') and a set of states C_i (PS)

E_i (PS, NS') = C_i (PS) • Δ_s (PS, NS')

Intersect set of given states with Δ_s

C'_{i+1} (NS') = S_{PS} E_i

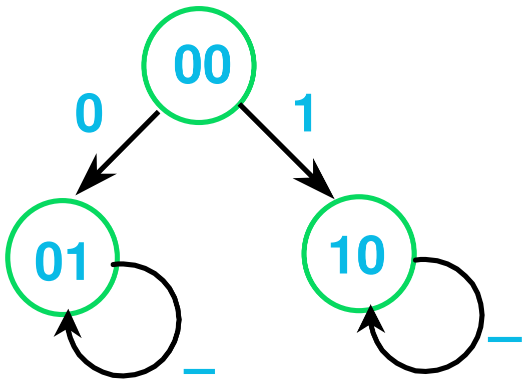
smooth away present state lines from E_i

C_{i+1}(PS) $\leftarrow$ C'_{i+1} (PS) $\cup$ C_i (PS)

$\downarrow$

converted support of C'_{i+1} from NS' to PS

Example



PI	PS	NS'	Δ
0	00	01	1
1	00	10	1
-	01	01	1
-	10	10	1

Δ_s (PS, NS')

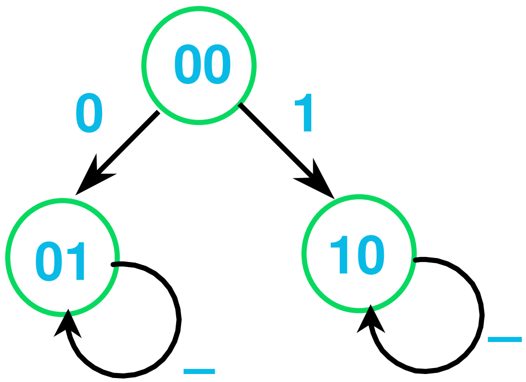
PS	NS'	Δ_s
00	01	1
00	10	1
01	01	1
10	10	1

$C_0 = 00$

$E_0 = C_0 \bullet \Delta_s$

PS	NS'	E_0
00	01	1
00	10	1

Example - 2



PS	NS'	E_0
00	01	1
00	10	1

$C'_1 = S_{PS}\, E_0 =$

NS'	C'_1
01	1
10	1

$C_1 = \{00, 01, 10\}$

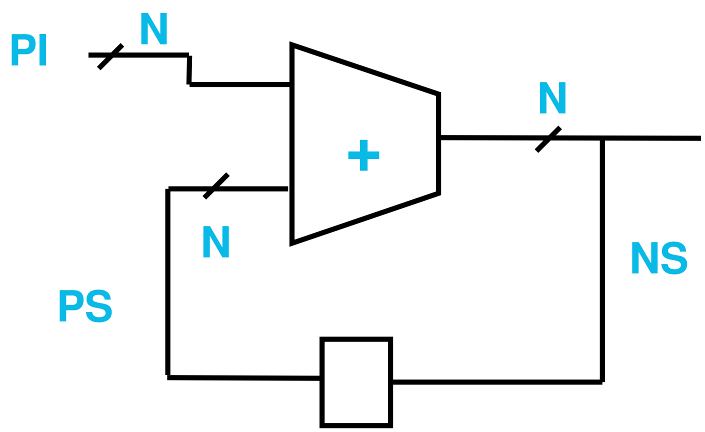
$E_1 = C_1 \bullet \Delta_S$

PS	NS'	E_1
00	01	1
00	10	1
01	01	1
10	10	1

$C'_2 = S_{PS}\, E_1 = \{01, 10\}$

$C_2 = \{00, 01, 10\}$

Datapath Circuit



- 2^N states, all non-equivalent
- Explicit enumeration impossible for large N
- Δ(PI, PS, NS') ROBDD size linear in N
- Reach every state from initial state 00...0 in one cycle
- Obtain $\boxed{1}$ ROBDD (if circuit is correct!)