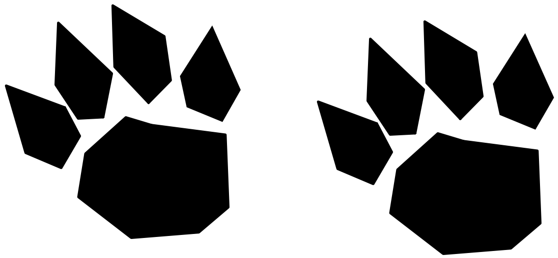
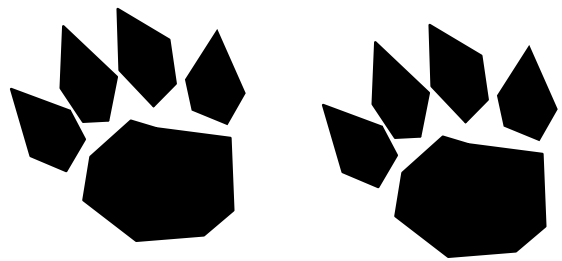
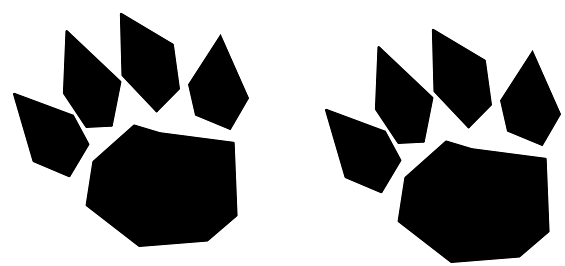


Steps

towards

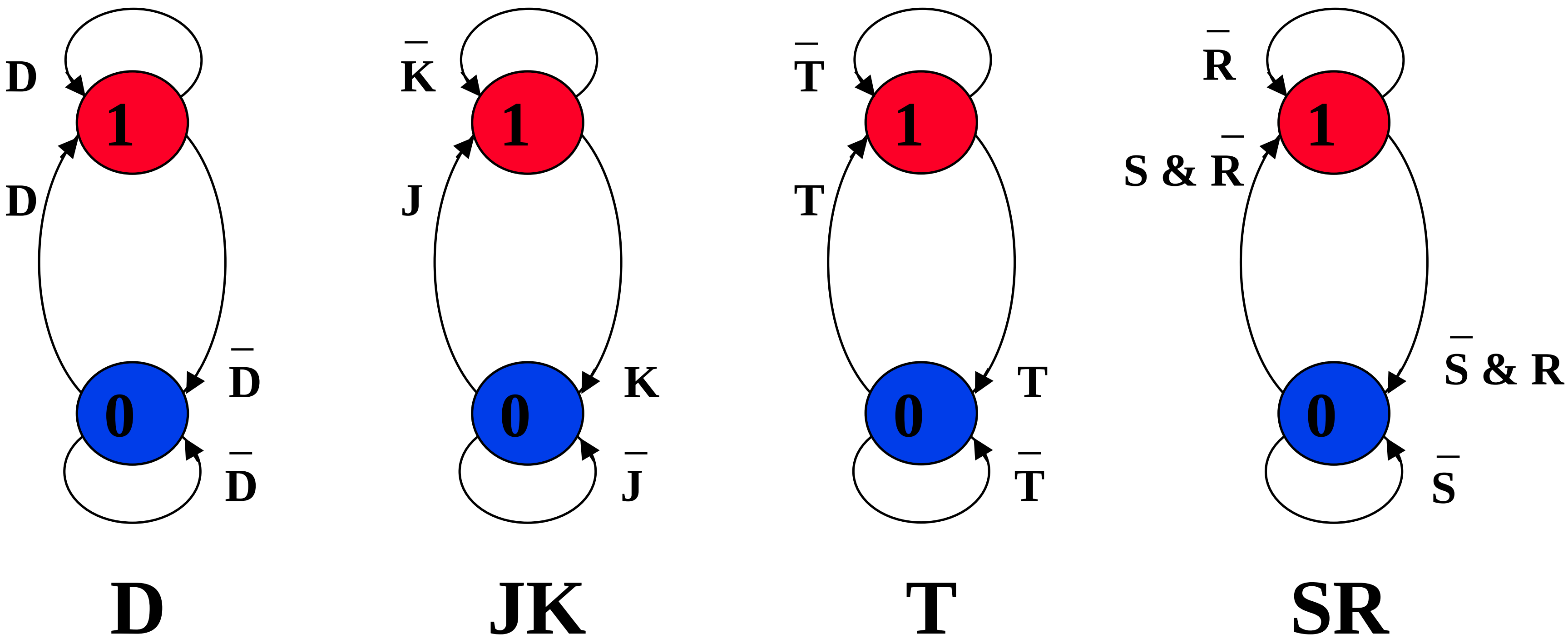


State assignment



Flip-Flops

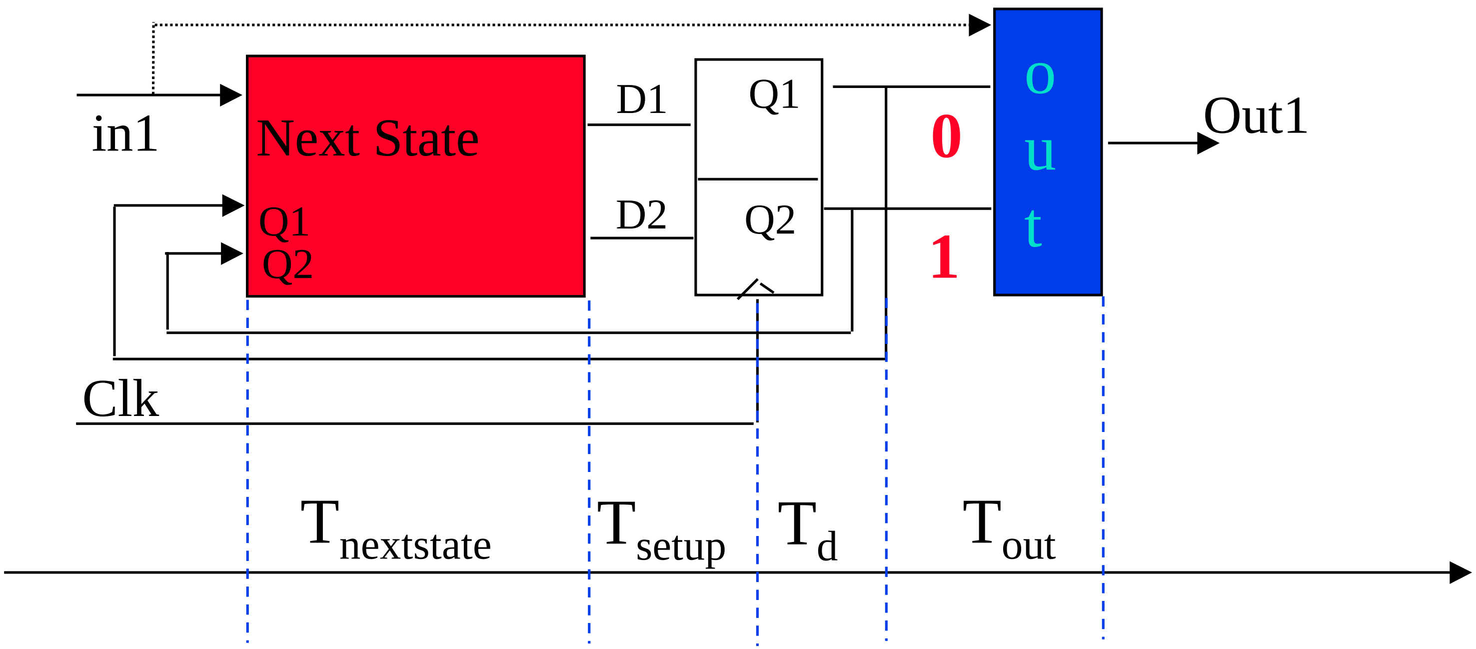
- FLIP-FLOPs are trivial FSMs
- Use state diagrams to remember flip-flops functions



FSM performance

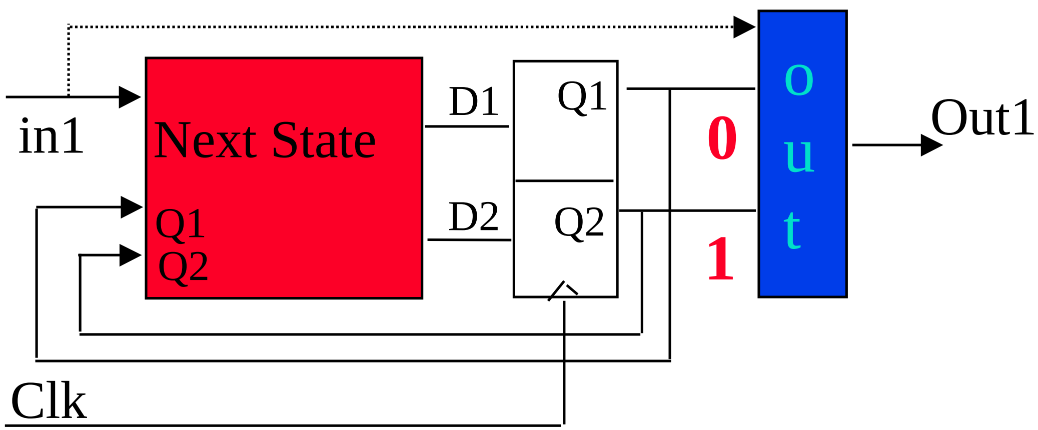
- Maximum frequency of operation is computed as :

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{T_{nextstate} + T_{setup} + T_d}$$

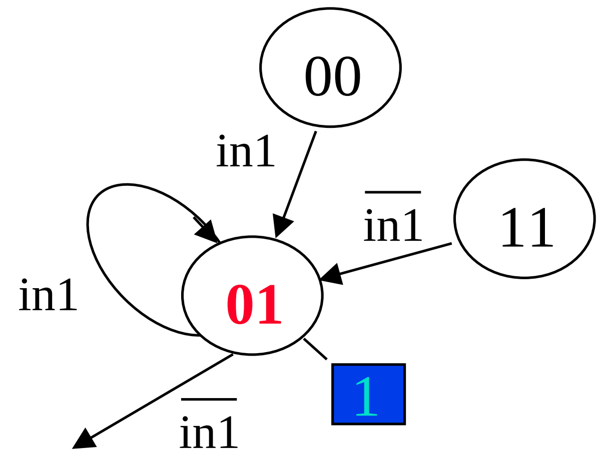


FSM Equations

- FSM next state and output logic can be easily derived by inspecting the state diagram.



Structure of a FSM with two F-Fs of D type

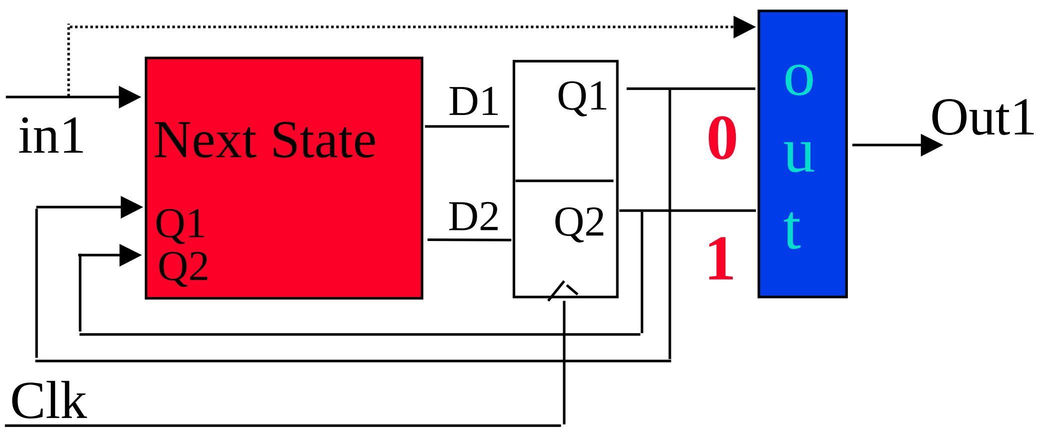


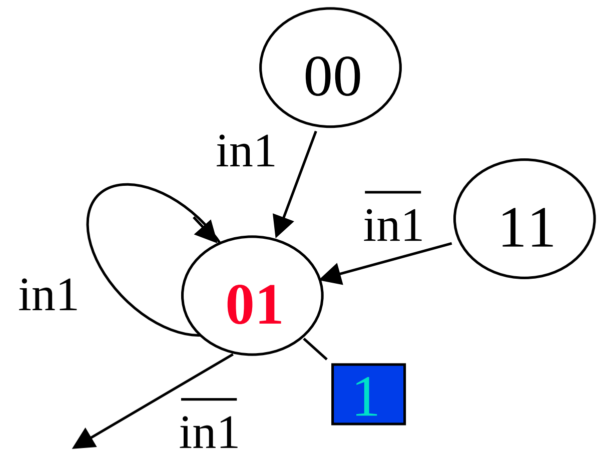
Sample part of a state diagram

What values of D1 and D2 will move the FSM to the state 01 ?

FSM Equations

- To move the FSM to state 01 the next state logic must produce '1' on D2 and '0' on D1.





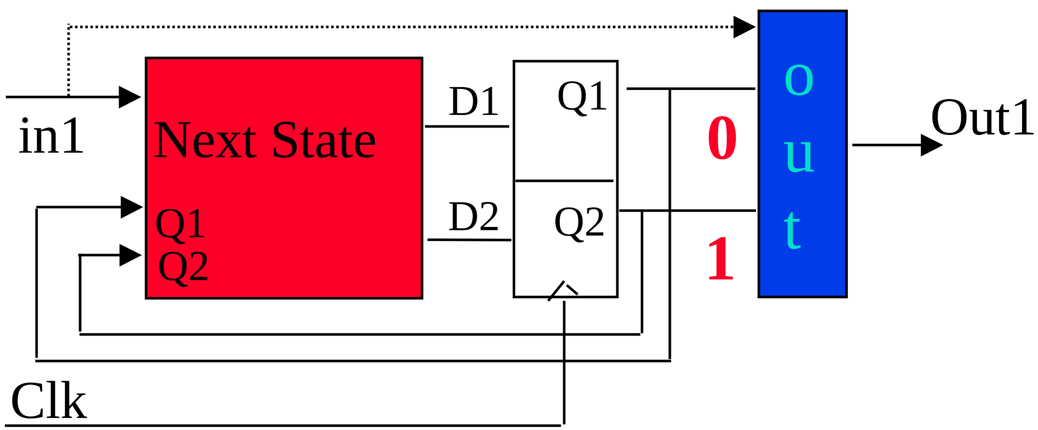
$$D1 = Q1^{+} =$$

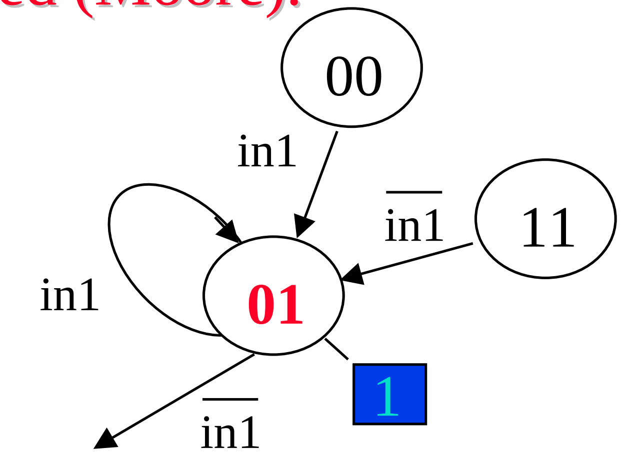
$$D2 = Q2^{+} = \text{in1} \; \& \; 00 \quad \# \quad \overline{\text{in1}} \; \& \; 11 \quad \# \quad \text{in1} \; \& \; 01 \quad \# \; \ldots$$

coming from state : 00 11 01

FSM Equations

- The output logic can be easily derived as a logical sum of all the states where ‘1’ on the output is produced (Moore).





$Out1 = \overline{Q1}\&Q2$ # ....

$D1 = Q1^{+} = \ldots$

$D2 = Q1^{+} = in1 \,\&\, 00 \;\#\; \overline{in1} \,\&\, 11 \;\#\; in1 \,\&\, 01 \;\# \ldots$

$\mathbf{D2 = Q1^{+} = in1\& \overline{Q1}\&\overline{Q2} \;\#\; \overline{in1}\& Q1\&Q2 \;\#\; in1\& \overline{Q1}\&Q2 \;\# \ldots}$

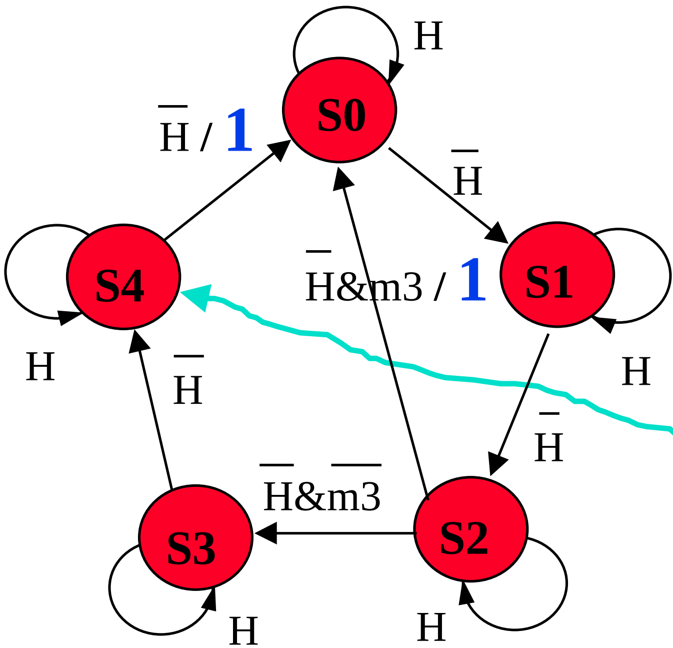
FSM Equations

- Next State Logic derivation algorithm for D type F-Fs

1. Choose a FF column in the state assignment table.
2. Find the first ‘1’ in this column and its corresponding state on the state diagram.
3. Write the product terms contributed by this state as the logical sum of the conditions on <u>all</u> incoming branches anded with the states they come from.
4. Find the next ‘1’ in this column and repeat the process 3 - 4 until all ‘1’ are used. You have obtained the next state Boolean function for the chosen FF.
5. Choose next FF column and repeat 2 - 5 until all FF are covered.

EXAMPLE

- Modulo 5/3 counter with Hold (Mealy machine)

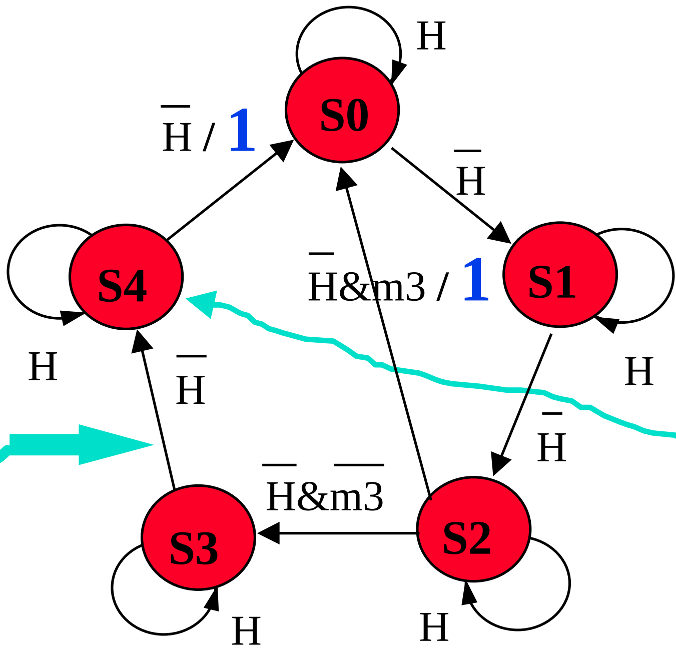


	Q0	Q1	Q2
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0

$Q0^{+} =$

Find the first ‘1’, find the corresponding state

EXAMPLE

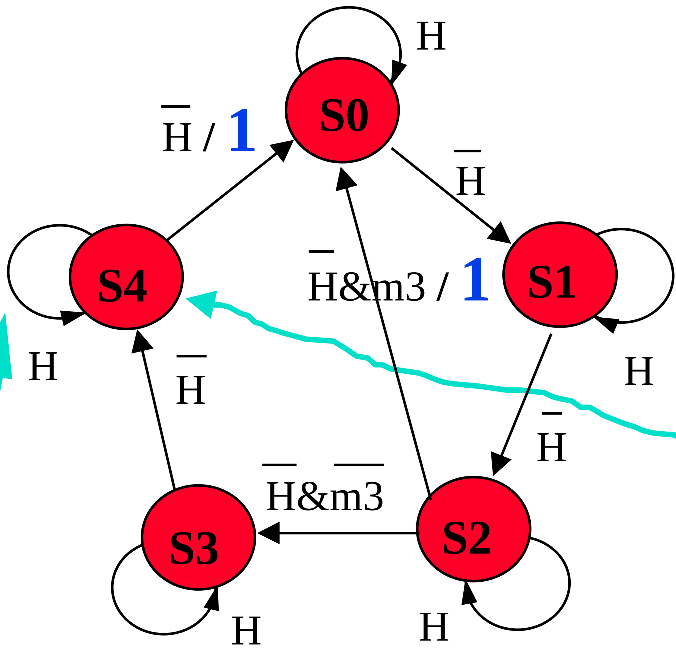


	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

$Q0^{+} = \overline{H}\&S3$ #

condition on an incoming branch AND the state it is coming from OR

EXAMPLE

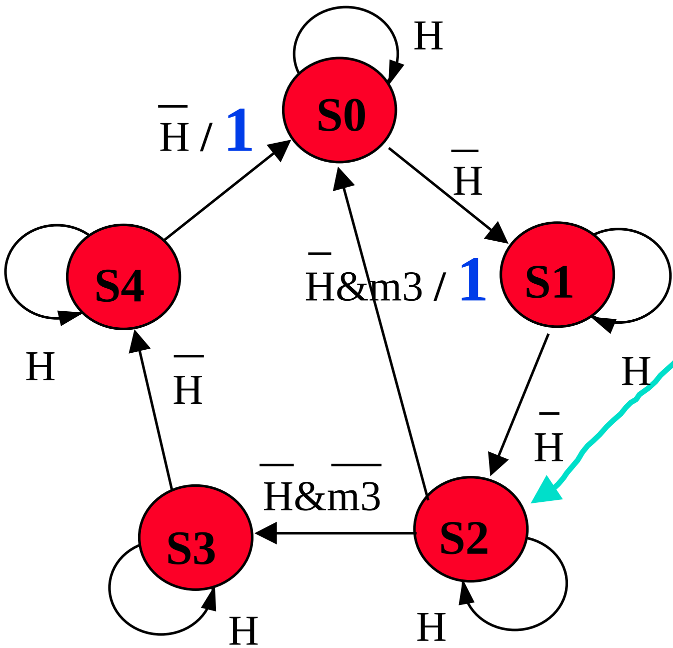


	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

$$Q0^{+} = \overline{H}\&S3 \# H\&S4$$

condition on an incoming branch AND
the state it is coming from OR

EXAMPLE



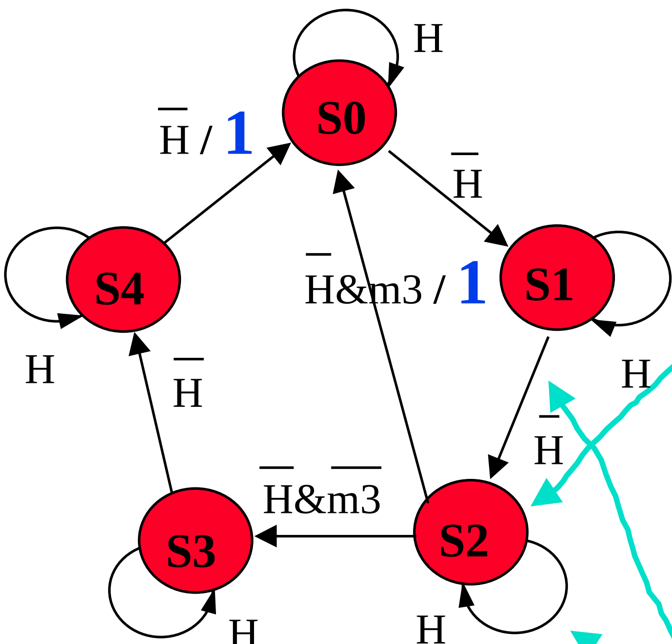
	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

$Q0^{+} = \overline{H}\&S3 \;\#\; H\&S4$

$Q1^{+} =$

Find the first ‘1’, find the corresponding state

EXAMPLE



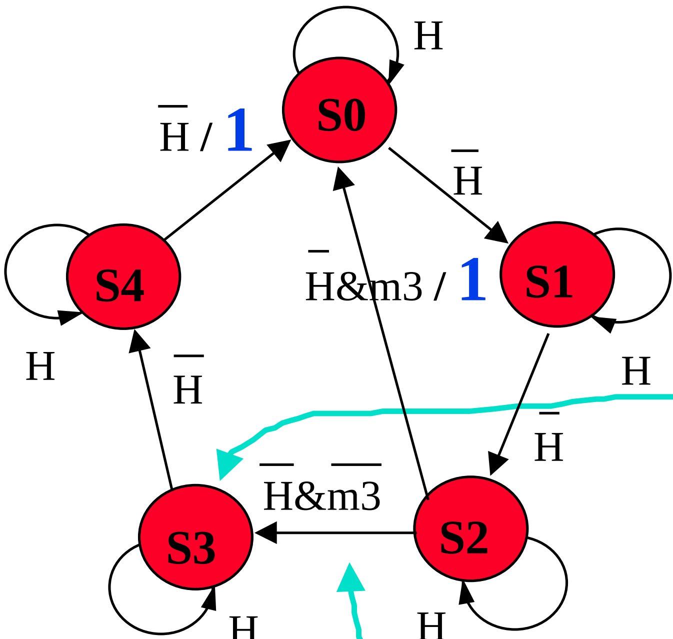
	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

$Q0^+ = \overline{H}\&S3 \# H\&S4$

$Q1^+ = \overline{H}\&S1 \# H\&S2 \#$

conditions on incoming branches ANDed
with the states they are coming from OR

EXAMPLE



	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

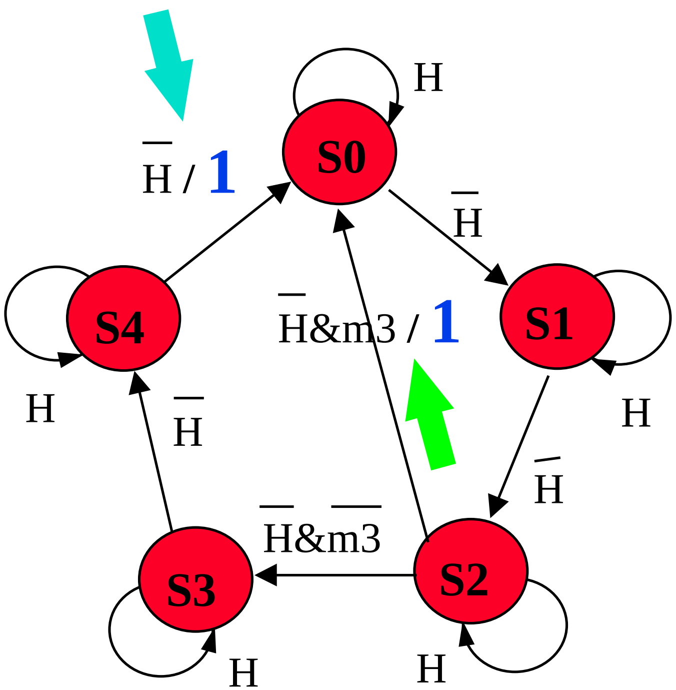
$Q0^{+} = \overline{H}\&S3 \# H\&S4$

$Q1^{+} = \overline{H}\&S1 \# H\&S2 \# \overline{H}\&\overline{m3}\&S2 \# H\&S3$

conditions on incoming branches ANDed with the states they are coming from OR

EXAMPLE

- The output logic for Mealy is derived as the logical sum of ‘1’ output conditions anded with the states they coming from



	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**0**	**0**	**1**
S2	**0**	**1**	**0**
S3	**0**	**1**	**1**
S4	**1**	**0**	**0**

$$Q0^{+} = \overline{H}\&S3 \;\#\; H\&S4$$

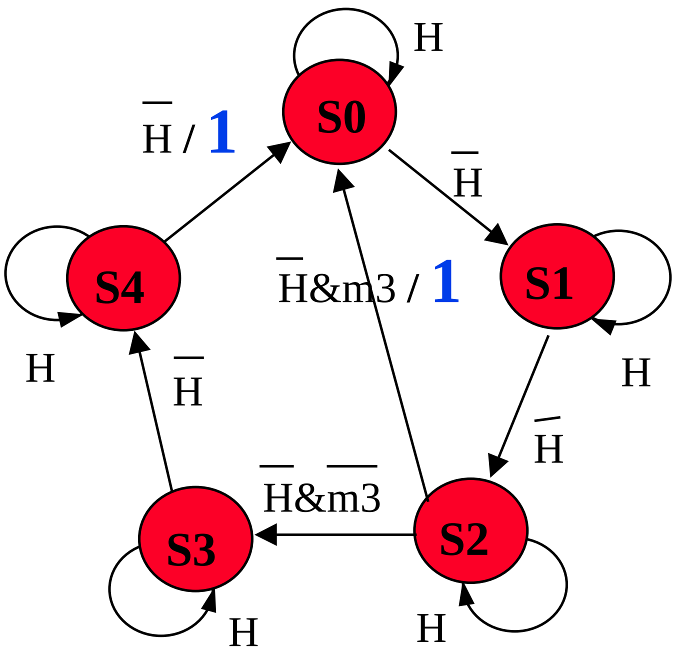
$$Q1^{+} = \overline{H}\&S1 \;\#\; H\&S2 \;\#\; \overline{H}\&\overline{m3}\&S2 \;\#\; H\&S3$$

$$Q2^{+} = \overline{H}\&S0 \;\#\; H\&S1 \;\#\; \overline{H}\&\overline{m3}\&S2 \;\#\; H\&S3$$

$$Out1 = \overline{H}\&S4 \;\#\; \overline{H}\&\overline{m3}\&S2$$

EXAMPLE

- Modulo 5/3 counter with Hold (Mealy machine)



	Q0	Q1	Q2
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0

$$Q0^{+} = \bar{H}\&S3 \# H\&S4$$

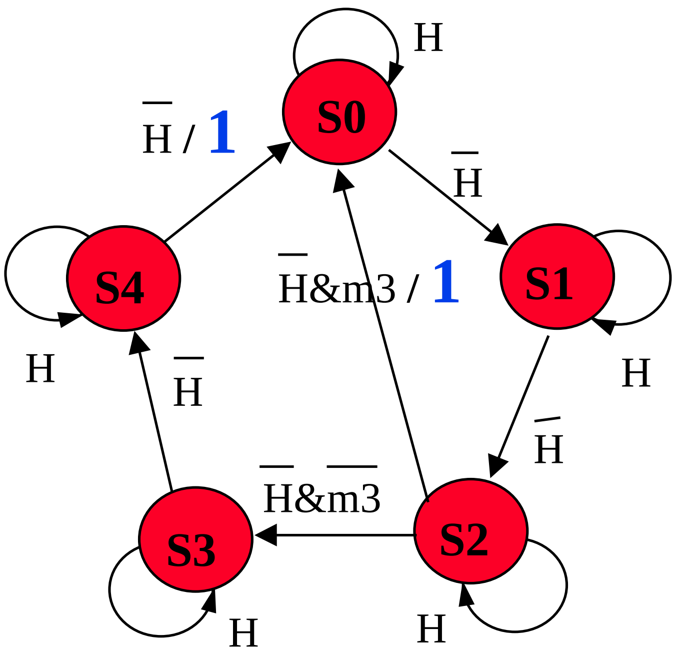
$$Q1^{+} = \bar{H}\&S1 \# H\&S2 \# \bar{H}\&\overline{m3}\&S2 \# H\&S3$$

$$Q2^{+} = \bar{H}\&S0 \# H\&S1 \# \bar{H}\&\overline{m3}\&S2 \# H\&S3$$

$$Out1 = \bar{H}\&S4 \# \bar{H}\&\overline{m3}\&S2$$

FSM state assignment

- Bad state encoding can result in larger next state logic.



	Q0	Q1	Q2
S0	**0**	**0**	**0**
S1	**1**	**0**	**1**
S2	**0**	**1**	**1**
S3	**0**	**1**	**0**
S4	**1**	**1**	**0**

$$Q0^{+} = \overline{H}\&S3 \,\#\, H\&S4 \,\#\, \overline{H}\&S0 \,\#\, H\&S1$$

$$Q1^{+} = \overline{H}\&S1 \,\#\, H\&S2 \,\#\, \overline{H}\&\overline{m3}\&S2 \,\#\, H\&S3 \,\#\, \overline{H}\&S3 \,\#\, H\&S4$$

$$Q2^{+} = \overline{H}\&S0 \,\#\, H\&S1 \,\#\, \overline{H}\&S1 \,\#\, H\&S2$$

$$Out1 = \overline{H}\&S4 \,\#\, \overline{H}\&\overline{m3}\&S2$$

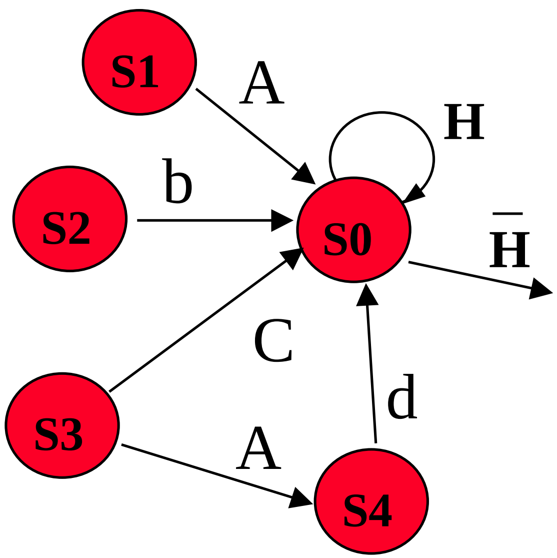
FSM state assignment by heuristics (educated guess)

- How does the next state and output logic depend on the codes assigned to the states ?
- Can we optimise logic better if we assign state codes in a smart way ?
- What is the smart way to assign the state codes ?
- Is it worth to try randomly and pick up the best code ?
- What are the guidelines to assign good codes ?
- When is it important to optimise the state codes ?
- NOTICE :
  operator NOT from now on is in two forms : ' $\overline{}$ ' and '!'

FSM state assignment

- **Golden Rules of good state encoding (for D FFs).**

1. States with most incoming branches should be assigned least '1's in their codes since they potentially contribute most product terms.



S0 = 1000

contributes

$Q0^{+}$ = ... # A&S1 # b&S2 # C&S3 # d&S4 # H&S0 # ...

S4 = 0001

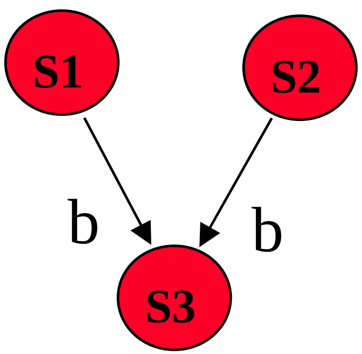
contributes

$Q3^{+}$ = ... # A&S3

FSM state assignment

- Golden Rules of good state encoding (for D FFs).

2. States with common next state on the same input condition should be assigned adjacent codes.



contributes

$$Q0^{+} = \ldots \# b\&S1 \# b\&S2 \#$$

$$= \ldots \# b\&(S1 \# S2) \# \ldots$$

S3 = 1000

S1 = 1100

S2 = 0100

adjacent codes

contributes

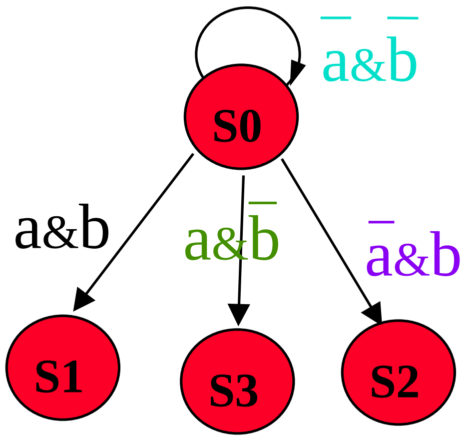
$$b\&(Q1\&Q2\&\overline{Q3}\&\overline{Q4} \# \overline{Q1}\&Q2\&\overline{Q3}\&\overline{Q4})$$

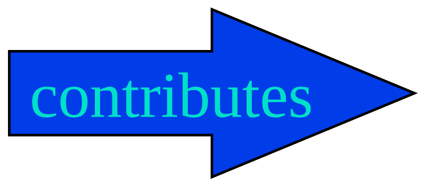
$$= b\& Q2\&\overline{Q3}\&\overline{Q4}$$

FSM state assignment

- Golden Rules of good state encoding (for D FFs).

3. Next states of the same state should be assigned adjacent codes according to adjacency of branch conditions.





adjacent codes

S0 = 0001

S1 = 1101

S2 = 0101

S3 = 1001

$Q0^{+}$ = a&b&S0 # a&!b&S0

= a&S0

$Q1^{+}$ = a&b&S0 # !a&b&S0

= b&S0

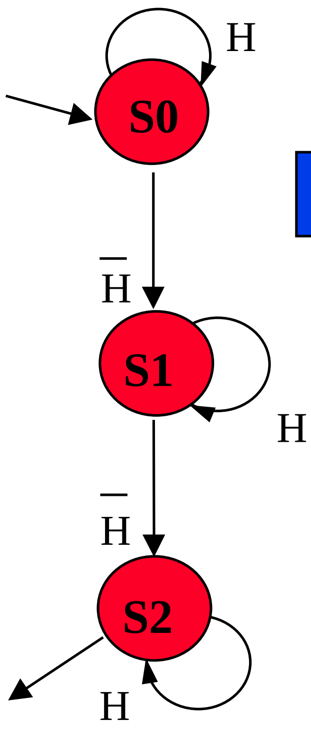
$Q3^{+}$ = a&b&S0 # a&!b&S0 # !a&b&S0 # !a&!b&S0

= S0

FSM state assignment

◆Golden Rules of good state encoding (for D FFs).

4. States that form a chain on the same branch condition should be assigned adjacent codes.



contributes

S0 = 001
S1 = 101
S2 = 111

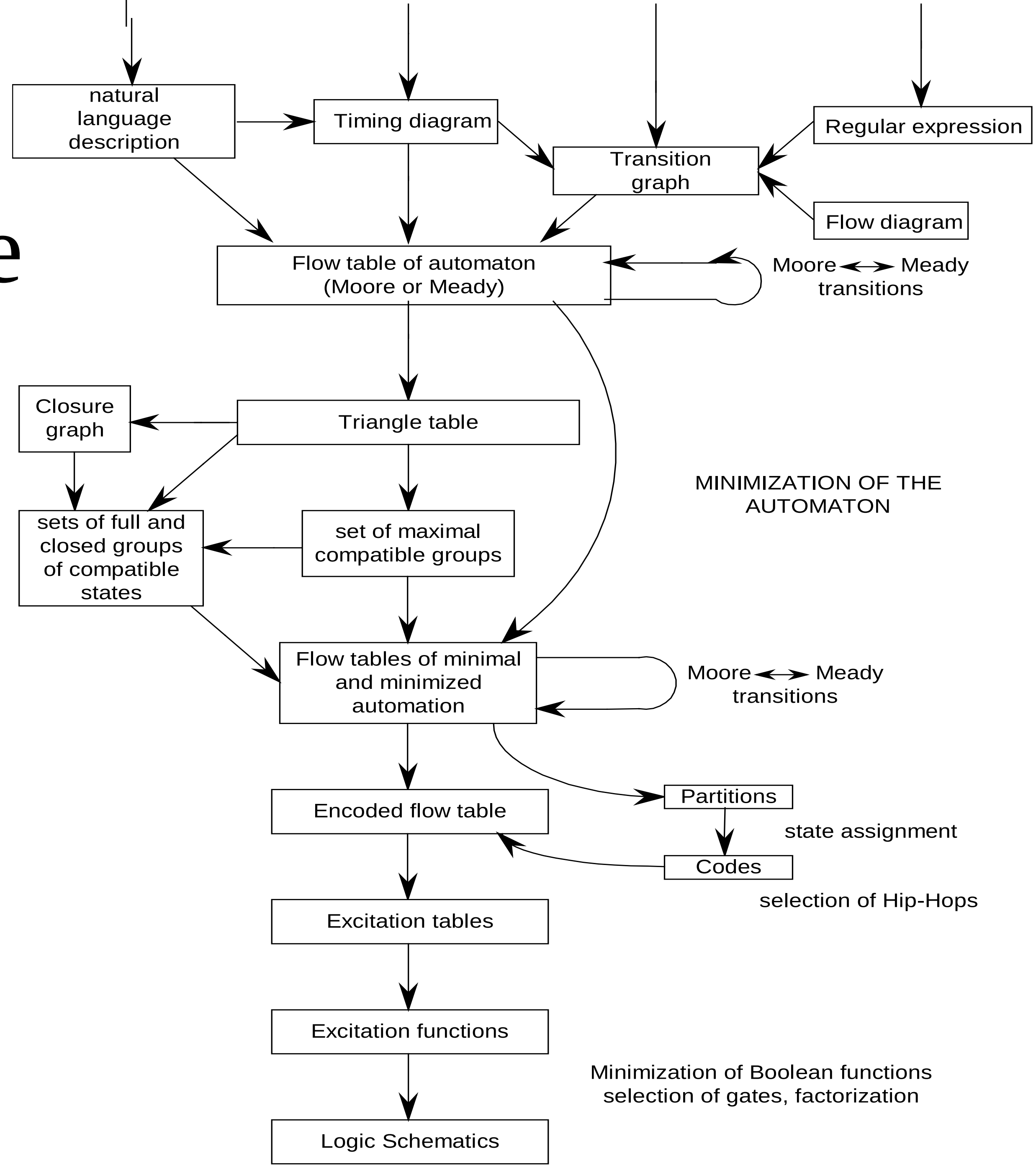
adjacent codes

$Q0^+$ = !H&S0 # H&S1 # !H&S1 # H&S2
= !H&(S0 # S1) # H&(S1 # S2)
= !H&!Q1&Q2 # H&Q0&Q2

$Q1^+$ = !H&S1 # H&S2
= !H&!Q0&Q1&Q2 #H&Q0&Q1&Q2

$Q2^+$ = H&S0 # !H&S0 # H&S1#!H&S1 # H&S2
= (S0 # S1) # H&S2
= !Q1&Q2 # H&Q0&Q1&Q2

Design Flow of Finite State Machine design



What have we learnt?

- Flip-Flops are trivial FSMs.
- Next State and Output Logic of FSMs can be easily derived by inspection of the State Diagram.
- State assignment can be performed by applying simple heuristics.
- State assignment is important since it can lead to substantial savings of next state and output logic.
- There are several methods of state assignment.

What else have we learnt?

- Inputs, states, and/or outputs can be encoded.
- Partition-based assignment methods give very good results with special properties but are hard computationally
- Partition-based methods are linked to decomposition
- Hypercube-embedding methods are fast and can give good results, but require usually computers
- Rule-based methods are not very good but allow for hand design
- Various encoding/decomposition methods can be combined for better results