Lecture 9
EE 421 / CS 425
Digital System Design

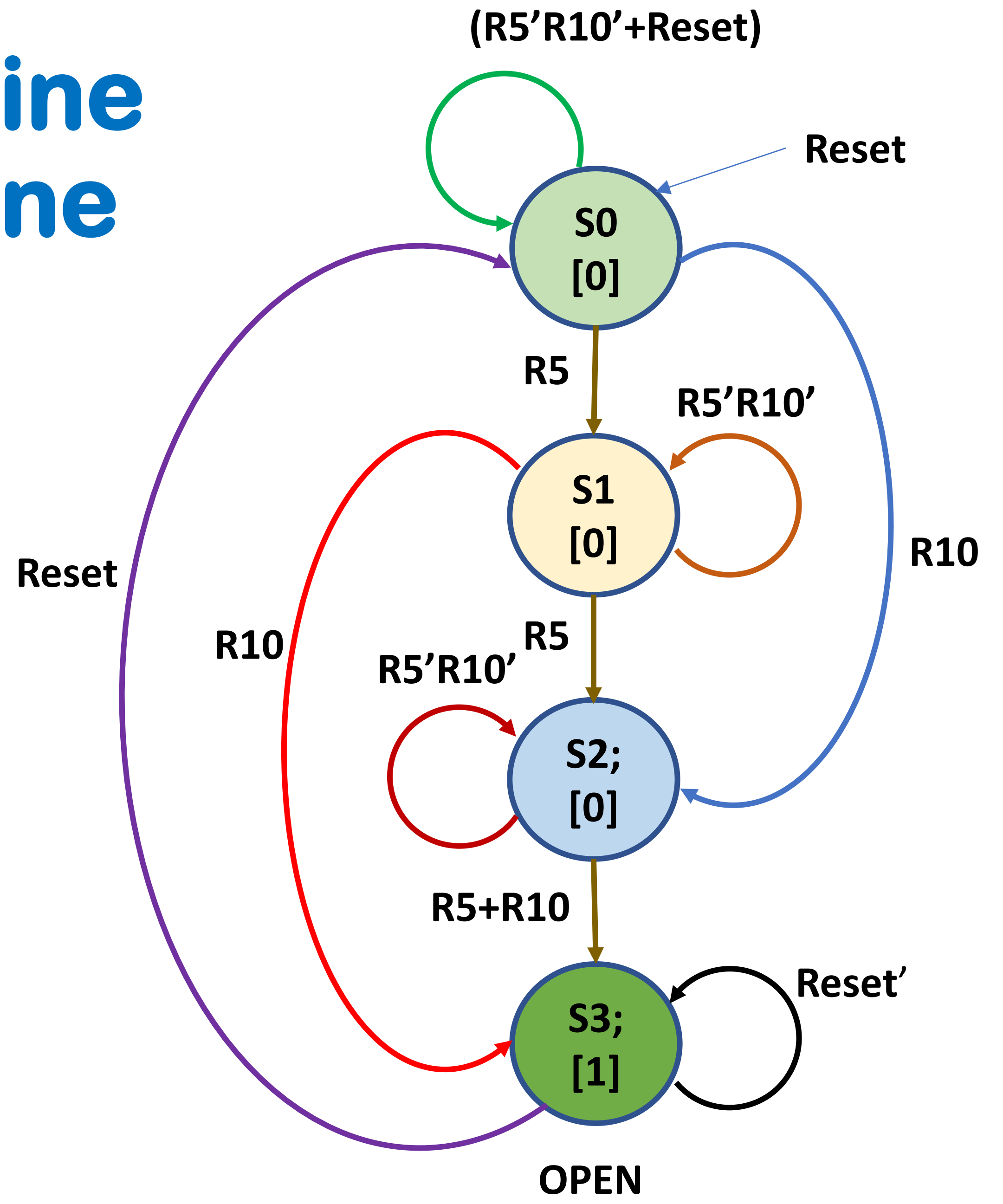
Fall 2025

Shahid Masud

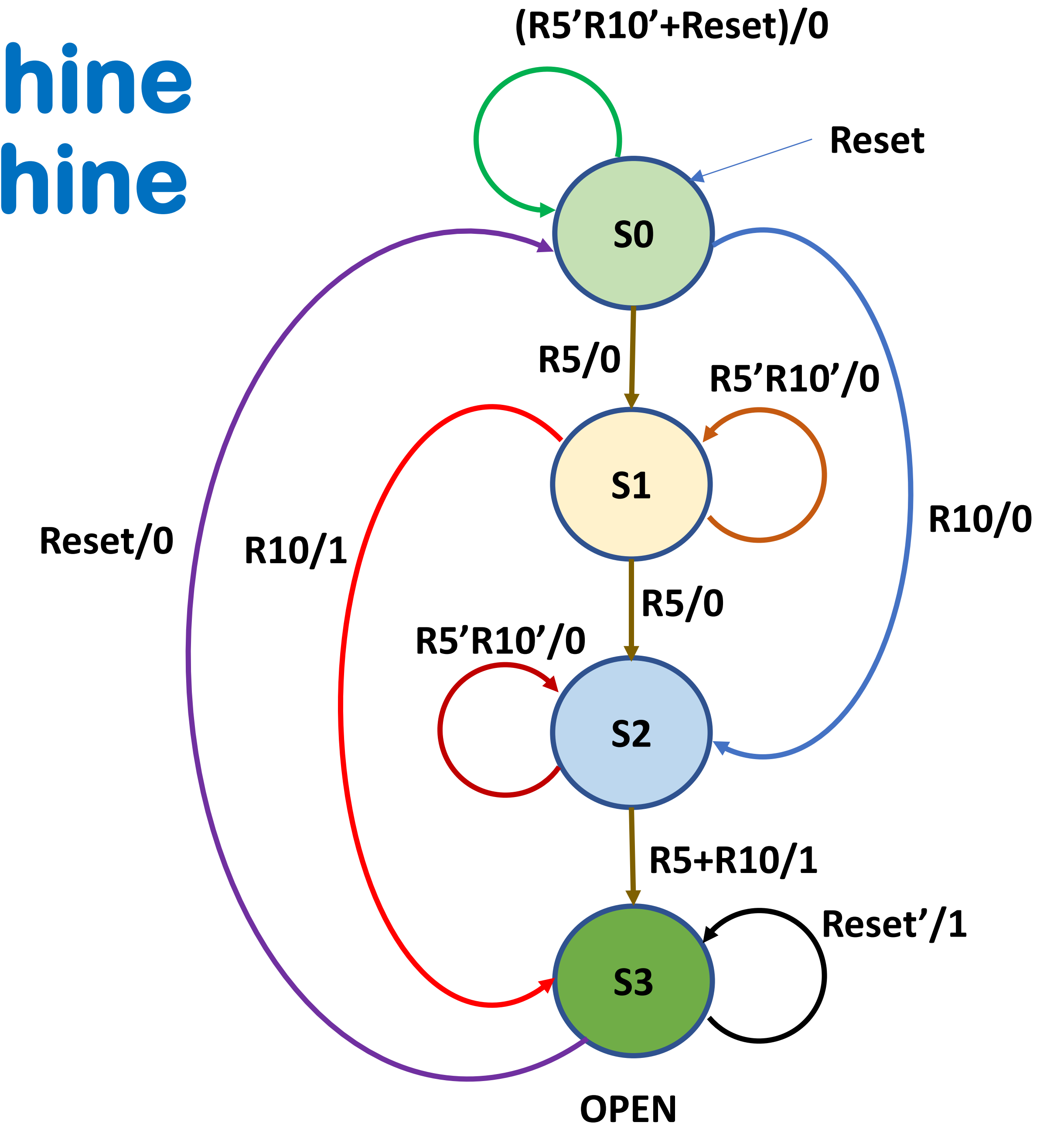
Topics

- (Leftover) Example of Vending Machine
- State Minimization
- Row Matching Method
- Using Implication Charts
- **QUIZ 2 in next lecture**

Moore State Machine for Vending Machine



Mealy State Machine for Vending Machine



Some Comparison Moore vs Mealy

- Mealy machine **requires fewer states** to reach output in comparison with Moore machine
- Mealy machine is **more susceptible to glitches**
- Explicit **output values** are shown in Mealy machine associated with each transition
- Output **changes after state** is changed in Moore machine
- Output in Moore machine **depends upon state only**; inputs can steer the output towards a particular state that affects output
- Output **depends upon present state and the present value** at the input; thus, output **can change immediately** with the change in input, independent of synchronous clock.

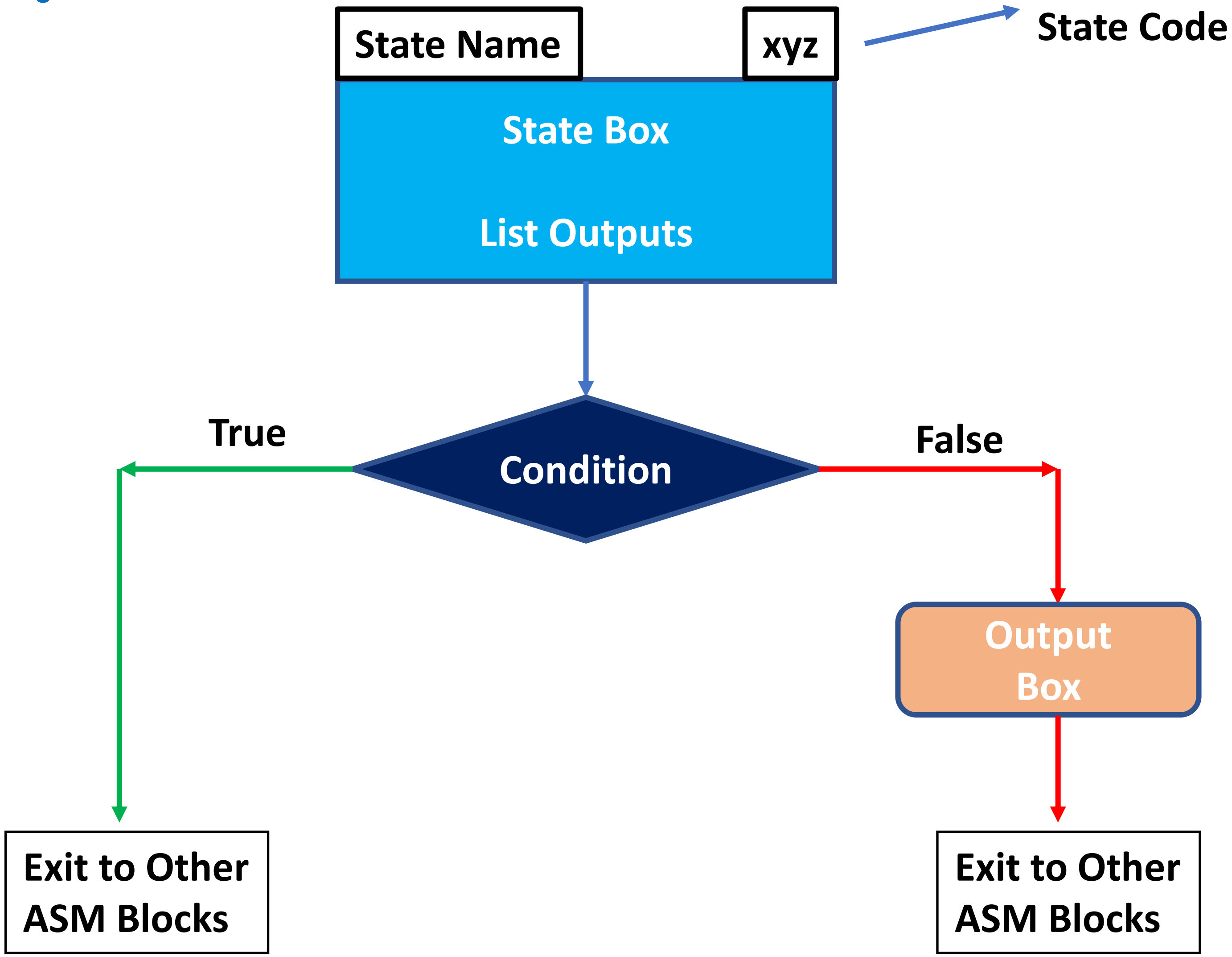
One Hot Encoding – one FF for each state

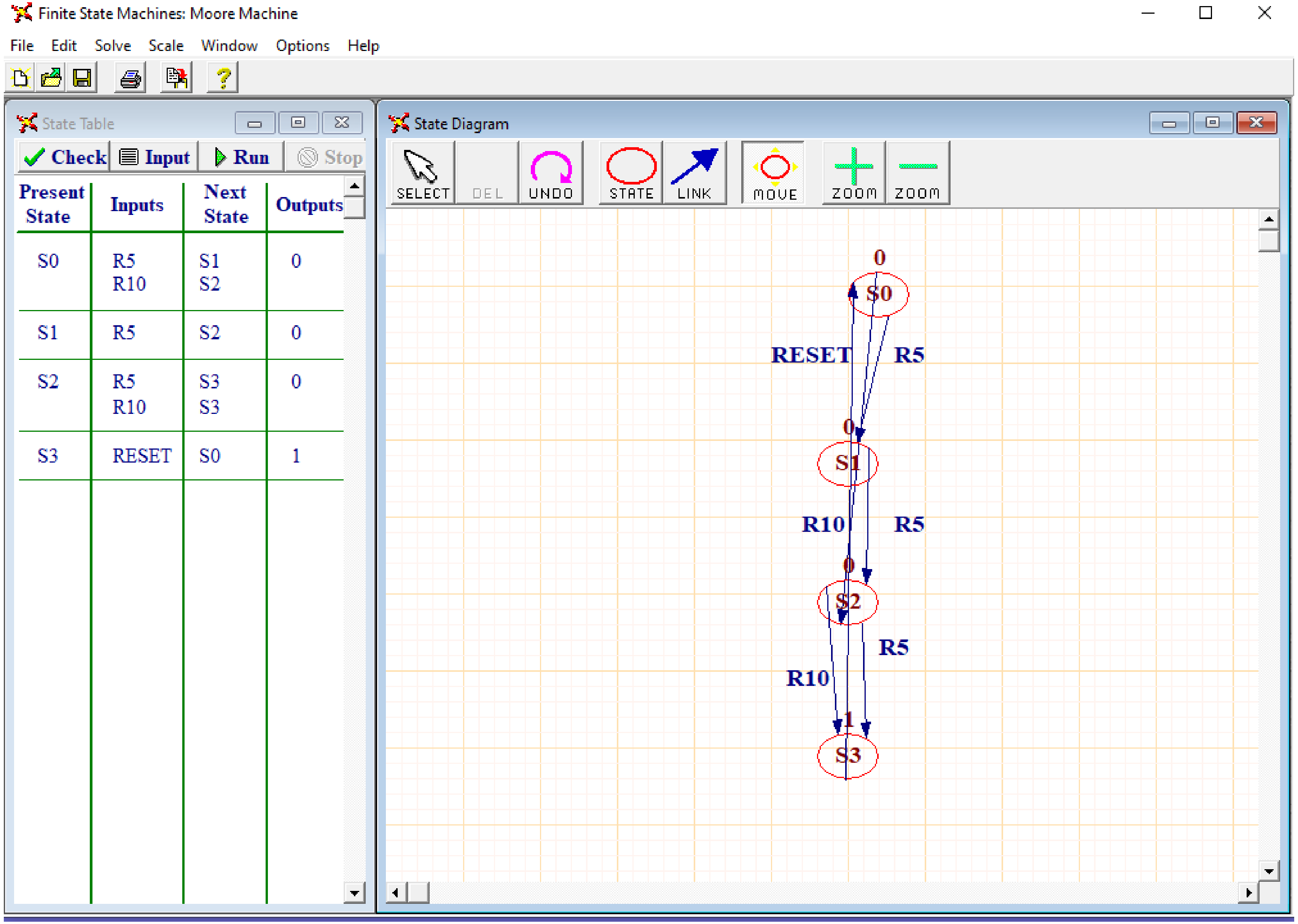
Present State				Inputs		Next State				Output OPEN
Q3	Q2	Q1	Q0	R10	R5	D3	D2	D1	D0	Y
0	0	0	1	0	0	0	0	0	1	
				0	1	0	0	1	0	
				1	0	0	1	0	0	
				1	1	X	X	X	X	
0	0	1	0	0	0	0	0	1	0	
				0	1	0	1	0	0	
				1	0	1	0	0	0	1
				1	1	X	X	X	X	
0	1	0	0	0	0	0	1	0	0	
				0	1	1	0	0	0	1
				1	0	1	0	0	0	1
				1	1	X	X	X	X	
1	0	0	0	X	X	1	0	0	0	1

D3 is directly the Output and Its State

The Design Becomes Simpler
Less Combinational Logic – At the Expense of Extra DFF

Algorithmic State Machine Description - ASMD





Present State	Inputs	Next State	Outputs
S0	R5 R10	S1 S2	0
S1	R5	S2	0
S2	R5 R10	S3 S3	0
S3	RESET	S0	1

Trying FSM implementation with WinlogiLab

Advantages of Minimum States

- Reduces number of logic gates and flipflops required for the implementation of state machine
- With fewer states, there are more don't care conditions into the next-state and output logic equations, making the implementation simpler.
- Simpler and less logic means shorter critical paths and higher achievable clock rate
- Fewer components also means shorter design time and lower manufacturing cost

State Minimization and Reduction

- State Reduction identifies and combines states that have equivalent behaviour

- Two states have equivalent behaviour iff:
  - for all input combinations, their outputs are the same, and
  - they change to the same or equivalent next state

State Reduction Algorithms

Begin with the symbolic state transition table.

We group together states that have the same state outputs (Moore Machine) or Transition outputs (Mealy Machine). These are potentially equivalent, since states cannot be equivalent if their outputs differ.

Next, examine the transitions to see if they go to the same next state for every input combination. If they do, the states are equivalent and can be combined into a renamed new state.

Then change all transitions into the newly combined states.

Repeat the process until no additional states can be combined.

Row Matching method for State Reduction

Use Case of a Four-Bit Sequence Detector

Specifications:

The machine has a single input X and a single output Z

The output is asserted after each 4-bit input sequence if the sequence contains binary Strings of "0110" or "1010"

The machine returns to "Reset" State after each 4-bit sequence

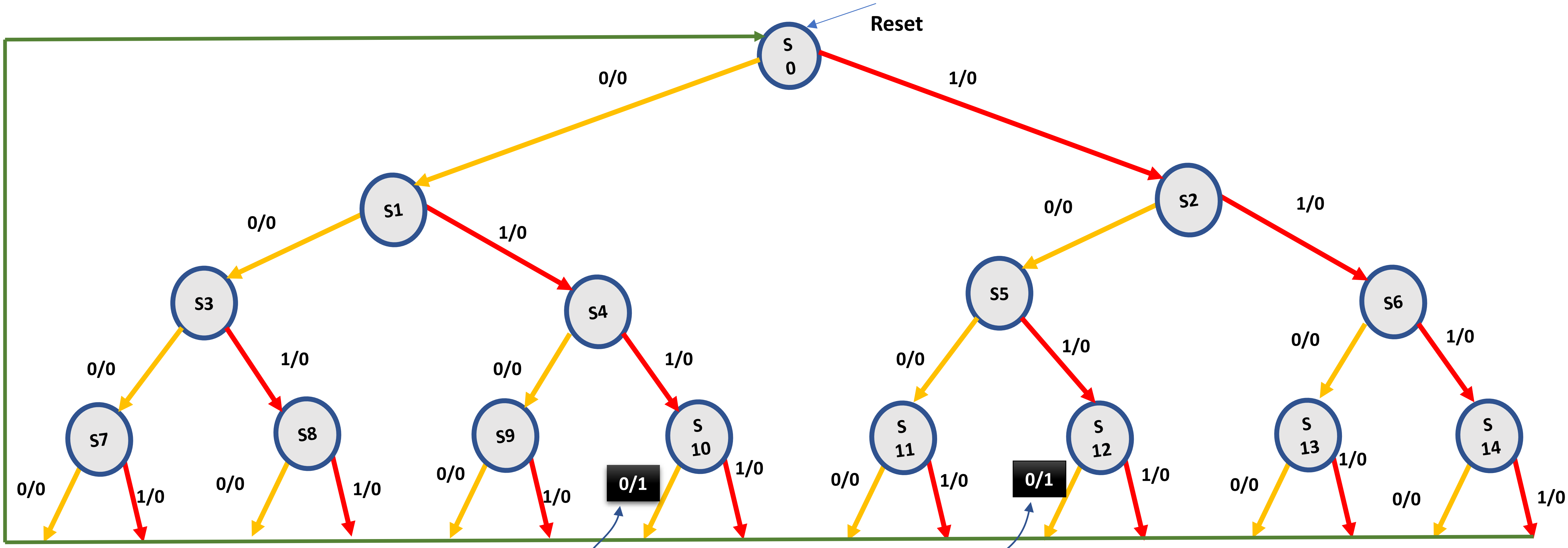
Assumptions:

Mealy Implementation

Example:

Input	X= 0010	0110	1100	1010	0011	…….	…….
Output	Z = 0000	0001	0000	0001	0000 ……	……	

Mealy State Machine for 4-Bit Sequence Detector



Look for Strings of “0110” or “1010” in serial input stream

Initial State Transition Table

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011	S10	S0	S0	1	0
100	S11	S0	S0	0	0
101	S12	S0	S0	1	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

Examine table to find rows with Identical next state and output values

Eg. We can combine S10 and S12; Both have same next state and same output

Merge S10 and S12 into one state “S10A” and make new table

Iteration:
States S7, S8, S9 and S11, S13, S14 have same Next states and same outputs; Hence these can be combined

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0
100	S11	S0	S0	0	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

Merge S7, S8, S9, S11, S13, S14 into one new State called S7A

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000, 001, 010, 100, 110, 111	S7A	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0

Replace S7, S8, S9, S11, S13, S14 with S7A in the table; Also S12 with S10A

Iteration:
States 00 and 11 have same Next State and Output (S3A)

States 01 and 10 have same Next State and Output (S4A)

Hence these are equivalent States

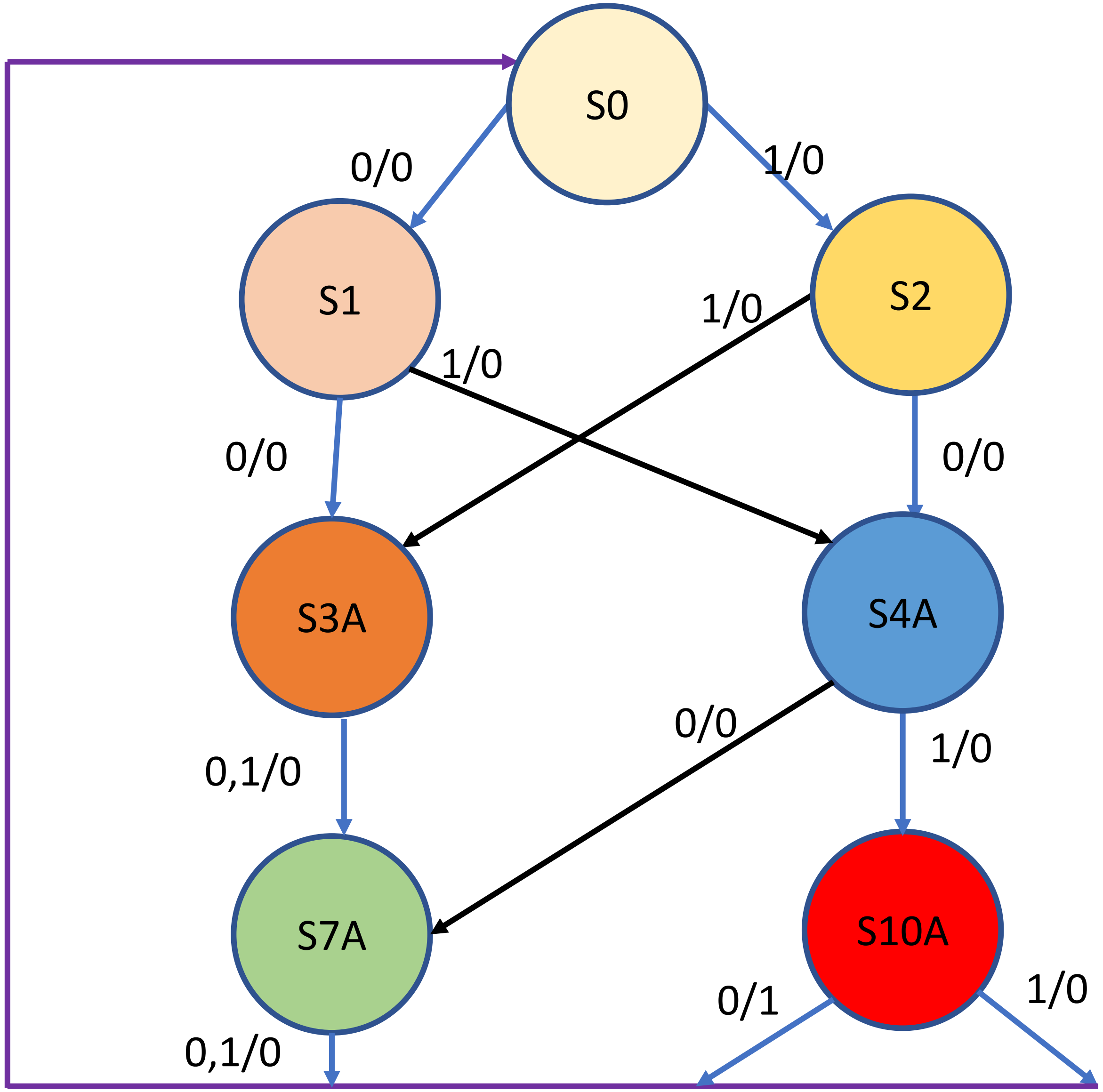
Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7A	0	0
00	S3	S7A	S7A	0	0
01	S4	S7A	S10A	0	0
10	S5	S7A	S10A	0	0
11	S6	S7A	S7A	0	0
000, 001, 010, 100, 110, 111	S7A	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0

Combine States S3 and S6 to S3A; Combine States S4 and S5 to S4A

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3A	S4A	0	0
1	S2	S4A	S7A	0	0
00, 11	S3A	S7A	S7A	0	0
01, 10	S4A	S7A	S10A	0	0
000, 001, 010, 100, 110, 111	S7A	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0

Final Reduced States are only 7:
S0, S1, S2, S3A, S4A, S7A, S10A

Final Reduced State Diagram



State Reduction using Implication Charts

- Graphical Technique for State Reduction
- Definition:
- Redundant State:
  - One state is equivalent to another (and hence redundant) if the state functions are in-distinguishable

- When all states that have (i) different next states and (ii) different outputs, have been identified, the remaining states are considered to be redundant

State Equivalence Theorem

- Two states SA and SB are equivalent iff for every possible input X sequence, the outputs are same and the next states are equivalent

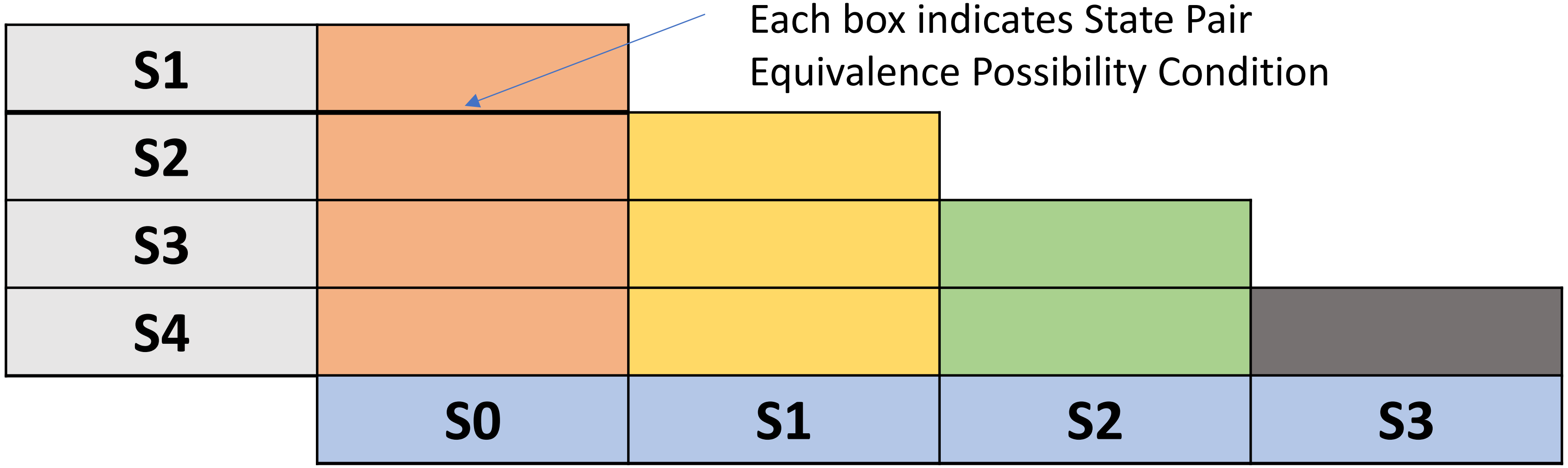
- Properties of Equivalent States:
- Symmetry: If SA=SB, then SB=SA
- Reflexivity: SA = SA for any state
- Transitivity: If SA=SB, and SB=SC, then SA=SC

Construction of Implication Charts

The Chart is constructed by listing all of the states except the last, along horizontal axis, and
Listing all of the states except the first, along the vertical axix
Intersection of the horizontal and vertical spaces indicates a possible state equivalene

Eg. Total States **S0 to S4**

Implies the possibility that S0 = S1
Each box indicates State Pair
Equivalence Possibility Condition



S1				
S2				
S3				
S4				
	S0	**S1**	**S2**	**S3**

Example of State Reduction using Implication Charts

Given the State Table as follows:

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

Total 6 states, S0 to S5

Filling the Implication Chart

- Place X in squares where outputs are different, as such states cannot be equivalent
- For other squares, we look at State Equivalence Pairs, i.e.:
  - S0 ≡ S2, iff S1 ≡ S3

  Thus we write S1,S3 in the square at intersection of S0, S2

  S0 ≡ S4, iff S2 ≡ S3, and S3 ≡ S5

  S1 ≡ S5, iff S2 ≡ S3

  S2 ≡ S4, iff S1 ≡ S5

  S3 ≡ S5, iff S1 ≡ S5, and S1 ≡ S2

Filled Implication Chart with conditions of equivalent states

S1	X				
S2	S1,S3	X			
S3	X	S1,S3	X		
S4	S2,S4 S3,S5	X	S2,S4 S1,S5	X	
S5	X	S1,S5 S2,S3	X	S1,S5 S1,S2	X
	S0	S1	S2	S3	S4

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

One by one, examine all table entries and refer to the state transition table

X is inserted where outputs of states is different
Check conditions of equivalence in respective squares

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

S1	X				
S2	S1,S3	X			
S3	X	S1,S3	X		
S4	S2,S4 S3,S5	X	S2,S4 S1,S5	X	
S5	X	S1,S5 S2,S3	X	S1,S5 S1,S2	X
	S0	S1	S2	S3	S4

To check for: {S1,S3}, {S1,S5}, {S2,S3}, {S1,S2}

To check for: {S1,S3}, {S1,S5}, {S2,S3}, {S1,S2}

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

In {S1,S3}:
Outputs are same for S1, and S3
In Next States, When X=1, S1 goes to S3 and S3 goes to S1
Hence S1 and S3 are equivalent

In {S1,S2}, output is different when X=1, hence these cannot Be equivalent, so this square will get ‘X’

In {S2,S3}, output is different when X=1, hence these cannot be equivalent, so this square will be ‘X’.
This means that S1 cannot be equivalent to S5 as it is dependent on S2 and S3 as Next State

Similarly, check all conditions in all squares

Updated Implication Chart

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

S1	X				
S2	S1,S3 ✓	X			
S3	X	S1,S3 ✓	X		
S4	S2,S4 S3,S5	X	S2,S4 S1,S5	X	
S5	X	S1,S5 S2,S3	X	S1,S5 S1,S2	X
	S0	S1	S2	S3	S4

This condition is established
So S0 and S2 are Equivalent States

S1 and S3 depend on themselves
Hence S1 and S3 are Equivalent states

Result:
S0 ≡ S2
And
S1 ≡ S3

Updated further - Implication Chart

This condition is established
So S0 and S2 are Equivalent States

S1 and S3 depend on themselves
Hence S1 and S3 are Equivalent states

S1	X				
S2	S1,S3	X			
S3	X	S1,S3	X		
S4	X	X	X	X	
S5	X	X	X	X	X
	S0	S1	S2	S3	S4

Result:
S0 ≡ S2
And
S1 ≡ S3

Mark **X** where further Equivalence is not apparent

Updated State Table

Original table with six states

Present State	Next State		Output	
	When X=0	When X=1	When X=0	When X=1
S0	S4	S3	0	1
S1	S5	S3	0	0
S2	S4	S1	0	1
S3	S5	S1	0	0
S4	S2	S5	0	1
S5	S1	S2	0	0

Result:
S0 ≡ S2
And
S1 ≡ S3

Only four states are left

	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
✓	S0	S4	S1	0	1
✓	S1	S5	S1	0	0
✗	~~S2~~	~~S4~~	~~S1~~	~~0~~	~~1~~
✗	~~S3~~	~~S5~~	~~S1~~	~~0~~	~~0~~
✓	S4	S0	S5	0	1
✓	S5	S1	S0	0	0

Practice Example: 4-Bit Sequence Detector

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3A	S4A	0	0
1	S2	S4A	S7A	0	0
00, 11	S3A	S7A	S7A	0	0
01, 10	S4A	S7A	S10A	0	0
000, 001, 010, 100, 110, 111	S7A	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0

See if states can be reduced further:

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3A	S4A	0	0
1	S2	S4A	S7A	0	0
00, 11	S3A	S7A	S7A	0	0
01, 10	S4A	S7A	S10A	0	0
000, 001, 010, 100, 110, 111	S7A	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0

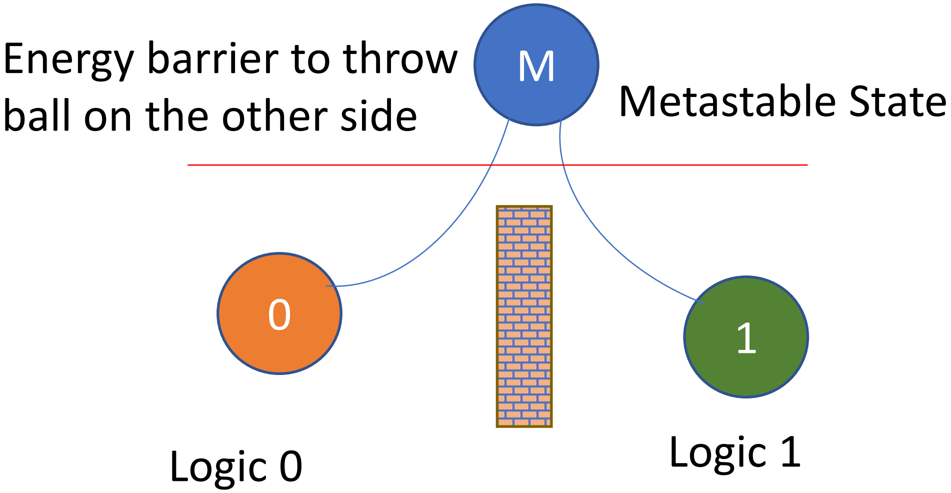
S1						
S2						
S3A						
S4A						
S7A						
S10A						
	S0	S1	S2	S3A	S4A	S7A

Filled Implication Chart

S1	🟧					
S2	🟧	🟨				
S3A	🟧	🟨	🟩			
S4A	🟧	🟨	🟩	⬜		
S7A	🟧	🟨	🟩	⬜	🟨	
S10A	🟧	🟨	🟩	⬜	🟨	🟦
	S0	S1	S2	S3A	S4A	S7A

Metastability, and Asynchronous Data Management

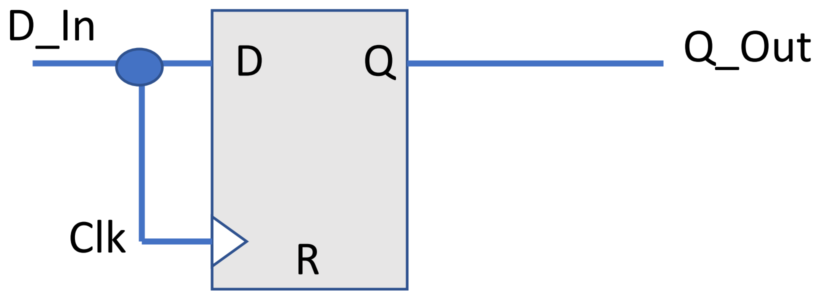
Objective – to reduce possible Metastability in capturing Asynchronous Input



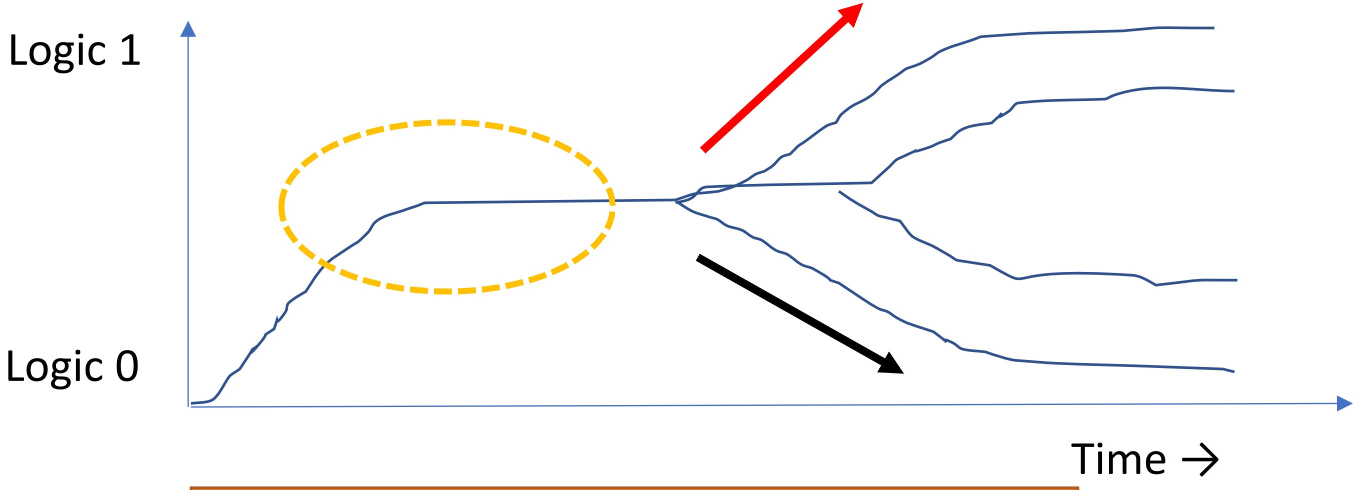
Synchronizer Failure: When flipflop hangs in a Metastable State for a long time (indefinitely)

Normally, the flipflop output would settle to a stable 0 or 1 state after some time

Output Behaviour with Metastability



DFF is connected to produce metastability As setup time is violated



Oscilloscope trace of metastable behaviour

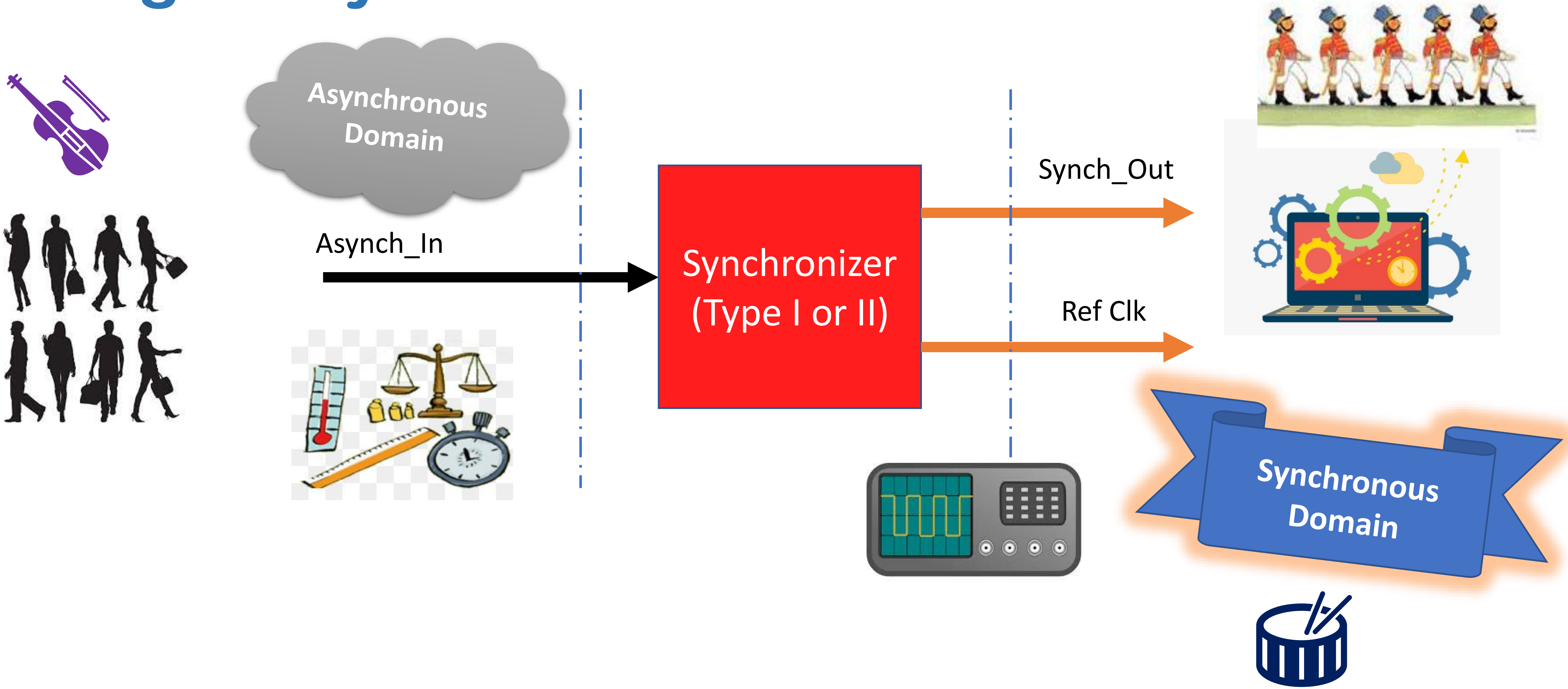
Eventually a stable state is reached

Problem occurs when flipflop is not stable within One clock period

Synchronizers

- **Asynchronous inputs are problematic as their transitions are not predictable**
- **High speed digital circuits rely on synchronizers to create a time buffer for recovering from a metastable event; thus reducing the possibility that metastability will cause circuit to malfunction**
- **An asynchronous signal should be synchronized by one synchronizer only. If not, then multiple synchronized signals could be present in the system and one of these could be driven into metastable state**

Asynchronous Inputs to a Synchronous Digital System



Asynchronous Domain

Asynch_In

Synchronizer (Type I or II)

Synch_Out

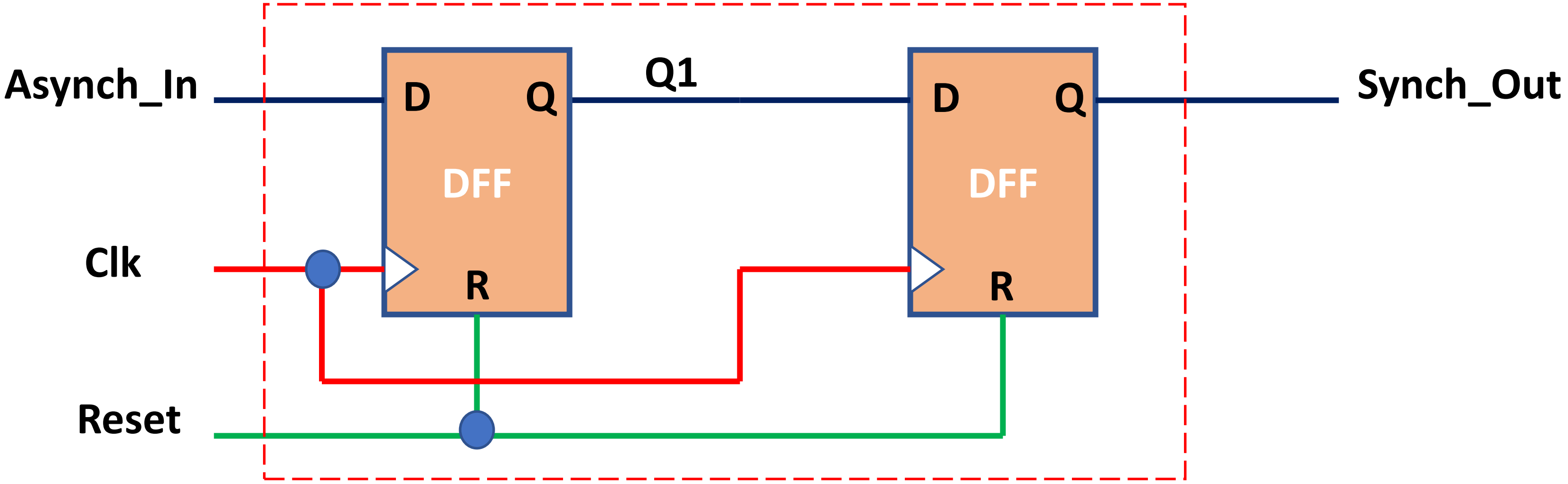
Ref Clk

Synchronous Domain

Synchronizer 1

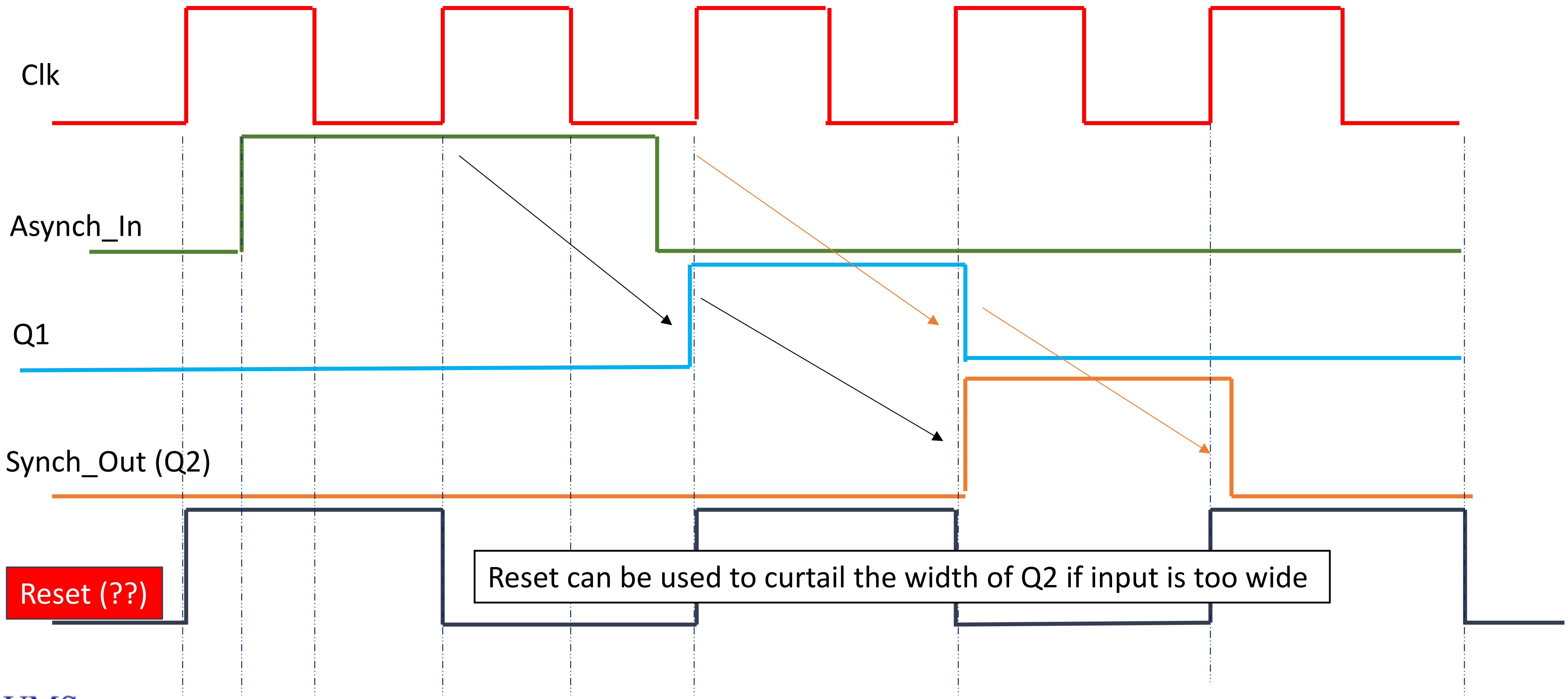
Condition: The width of asynchronous input pulse is greater than period of the clock

Synchronizer is a multistage shift register



Reset Input 'R' is used as control to bring Synch_Out back to '0', as required

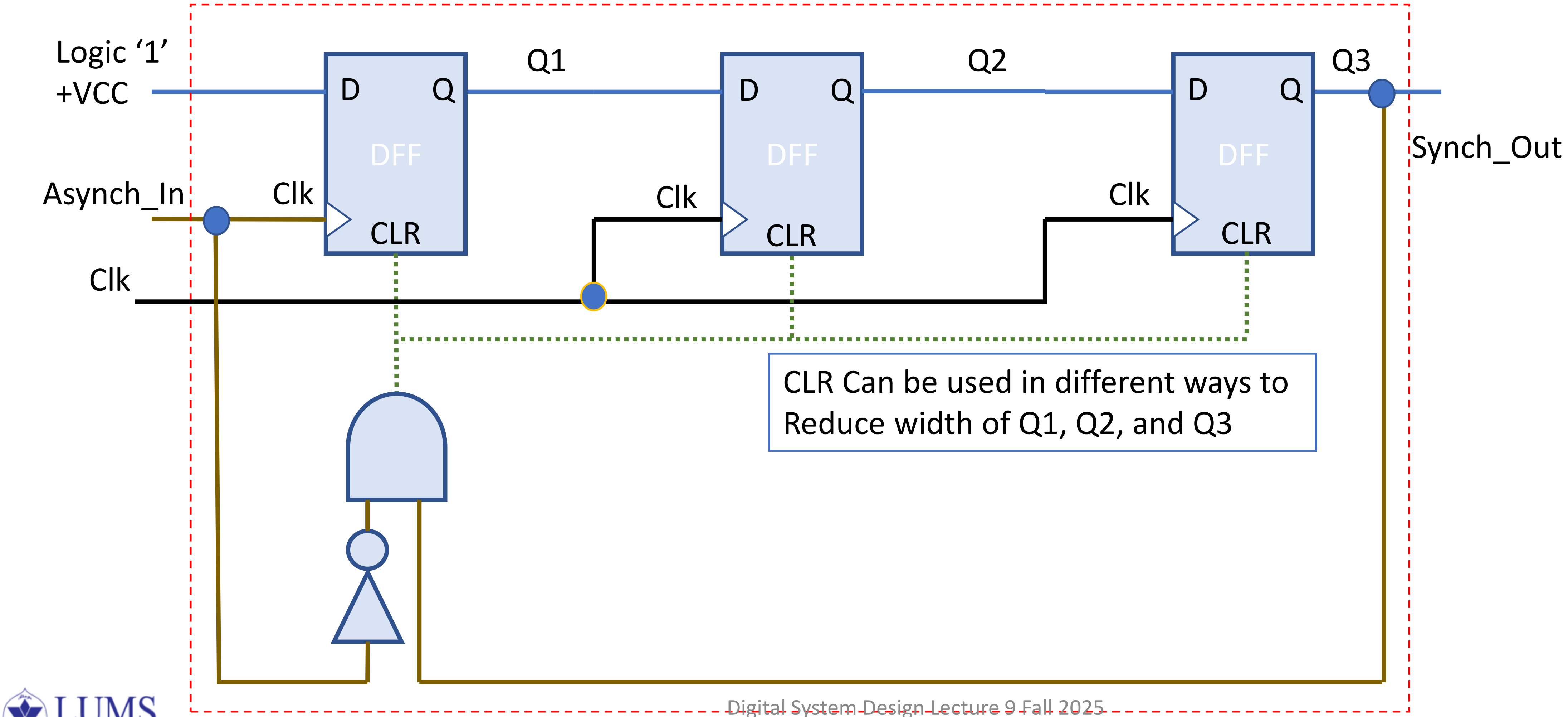
Timing Diagram – Synchronizer 1



Reset can be used to curtail the width of Q2 if input is too wide

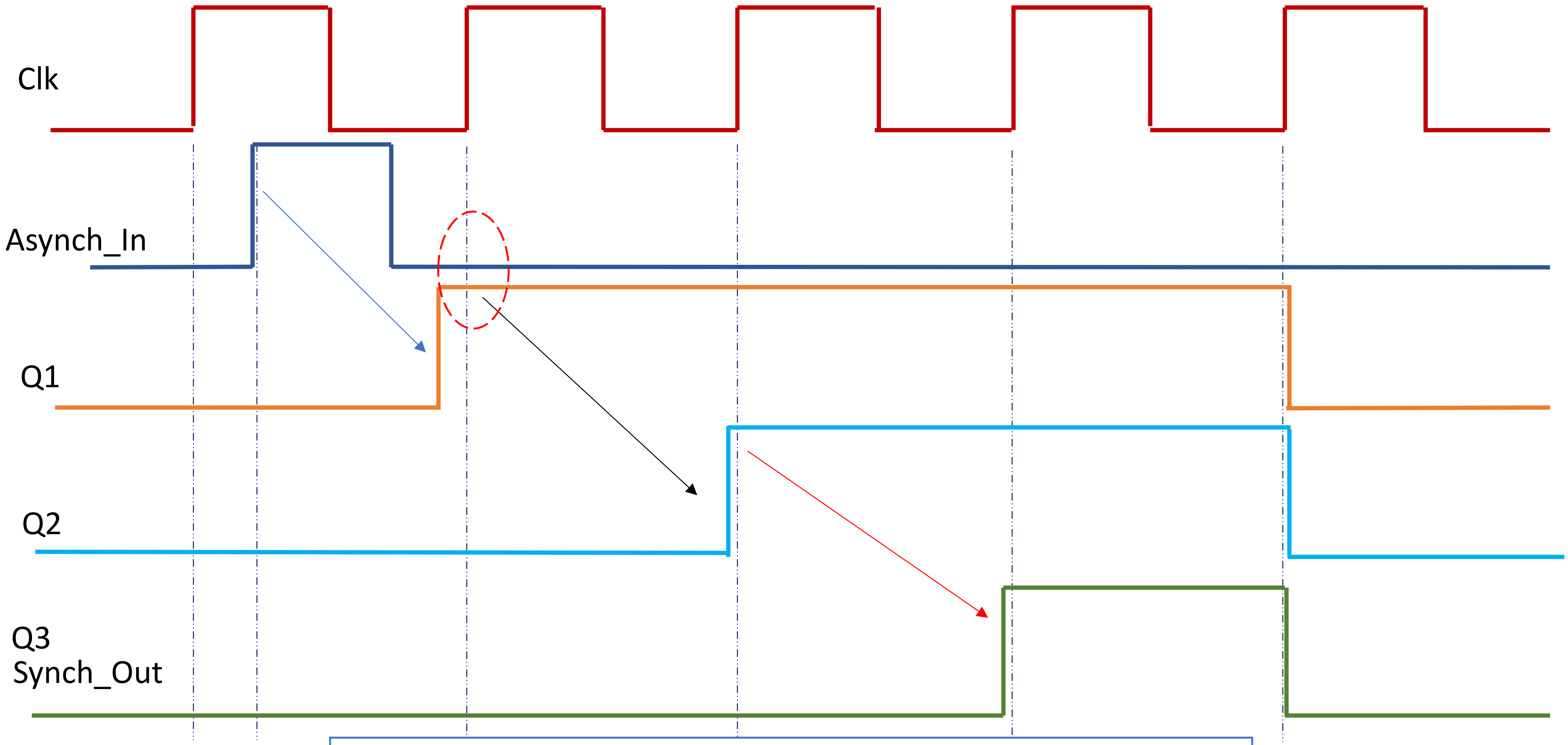
Synchronizer 2

Condition: Width of the asynchronous input pulse is less than the period of the clock



CLR Can be used in different ways to Reduce width of Q1, Q2, and Q3

Timing Diagram – Synchronizer 2

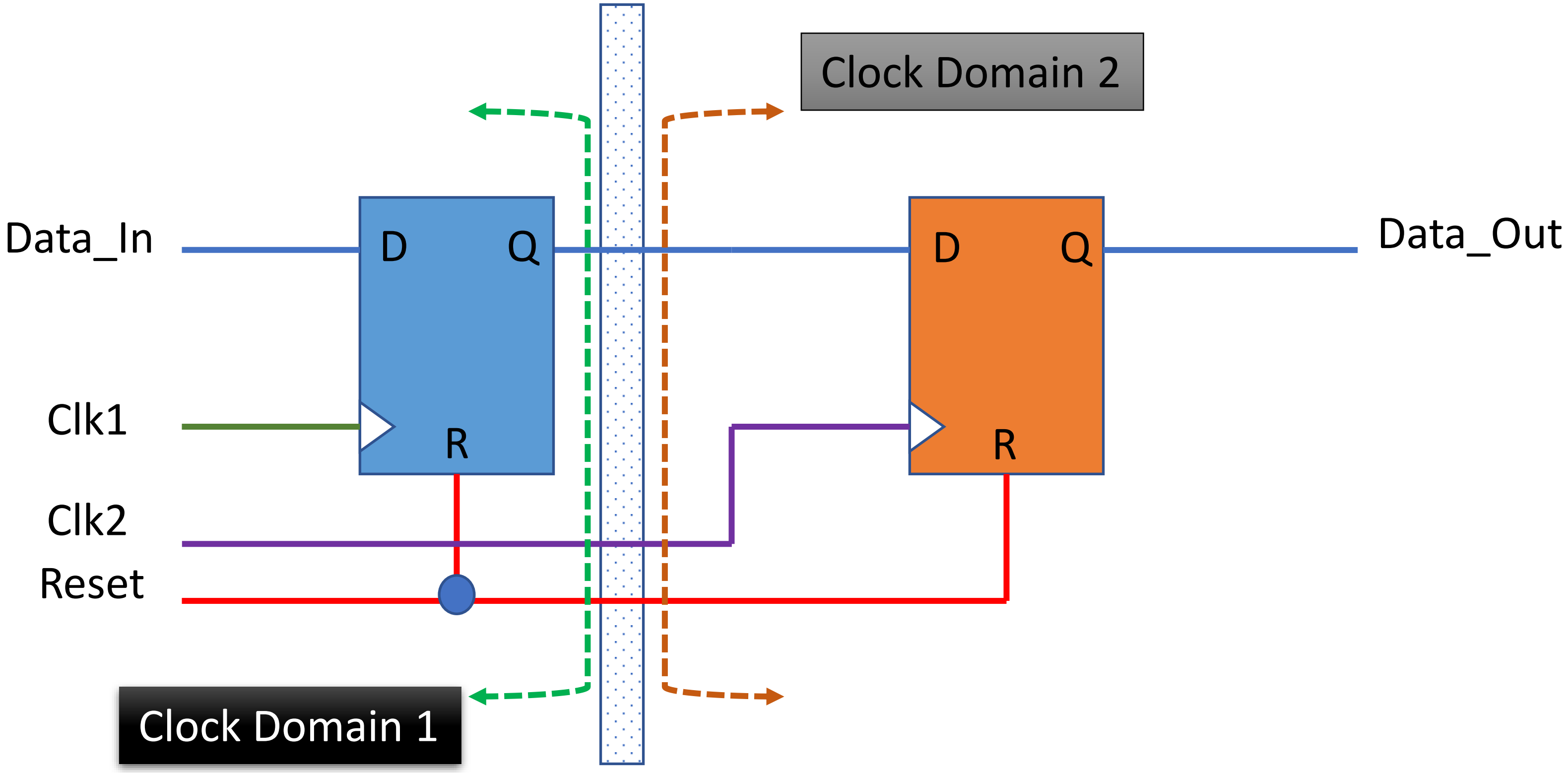


CLR input at DFF1 is used to bring back Q1 and then Q2 to Zero

Self-Timed and Speed Independent Circuits

- Having a completed Synchronous system is at times too challenging for a complex and fast digital system
- The limiting problem becomes how to distribute a single global clock without introducing intolerable clock skew
- The alternate is to partition the digital system into locally clocked pieces that communicate with each other using delay-insensitive signaling techniques (i.e. local clock for local communication)
- Each block proceeds at its own speed without the need for a global clock, synchronizing local communication whenever needed
- Usually a Request-Response Signalling method is employed

Data Reading across two Clock Domains



frequency of Clk1 is less than Clk2
Otherwise, the Synchronizer-2 is versatile and can be used here