IE1204 Digital Design

F12: Asynchronous Sequential Circuits (Part 1)

KTH Informations- och kommunikationsteknik

Masoumeh (Azin) Ebrahimi (masebr@kth.se)

Elena Dubrova (dubrova@kth.se)

KTH / ICT / ES

This lecture

- BV pp. 584-640

Asynchronous Sequential Machines

- An asynchronous sequential machine is a sequential machine without flip-flops
- Asynchronous sequential machines are constructed by analyzing combinational logic circuits with feedback
- **Assumption: Only one signal in a circuit can change its value at any time**

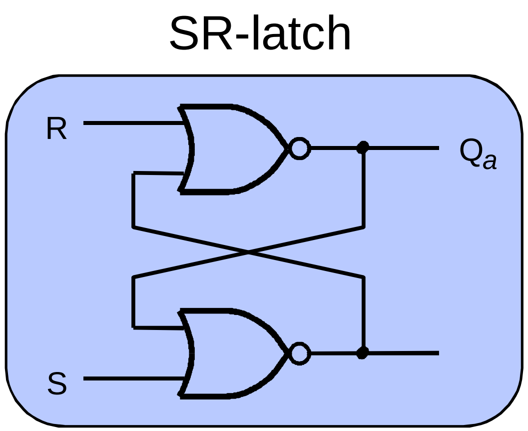
Golden rule

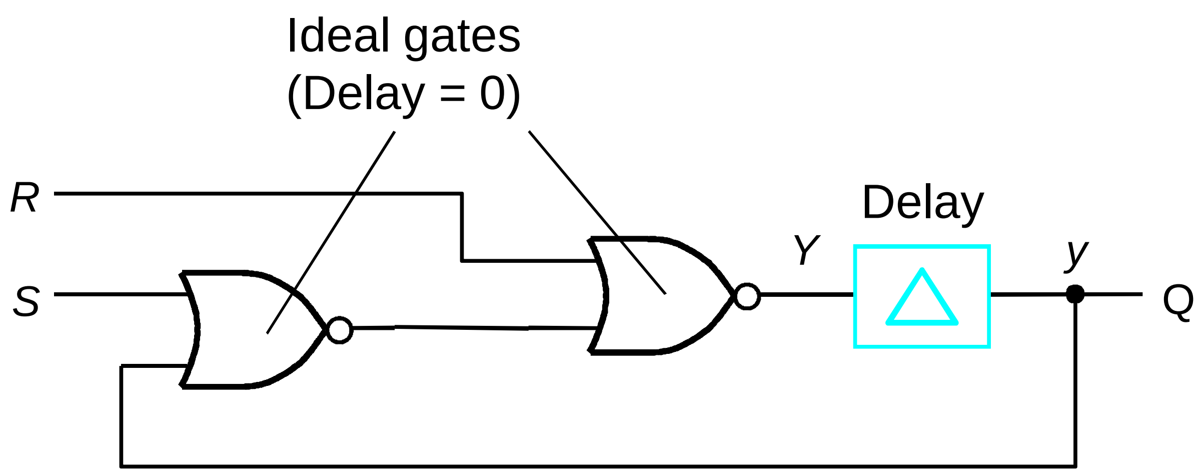
Asynchronous state machines

- Asynchronous state machines are used when it is necessary to keep the information about a state, but no clock is available
  - All flip-flops and latches are asynchronous state machines
  - Useful to synchronize events in situations where metastability is/can be a problem

Asynchronous sequential circuit: SR-latch with NOR gates

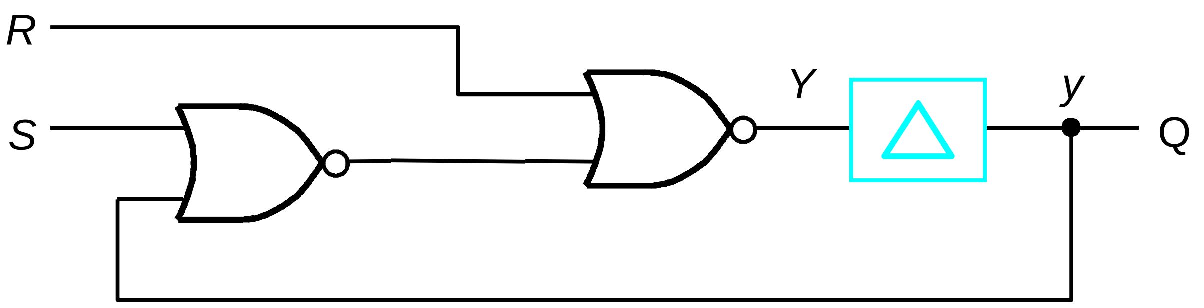
- To analyze the behavior of an asynchronous circuit, we use ideal gates and summarize their delays to a single block with delay Δ





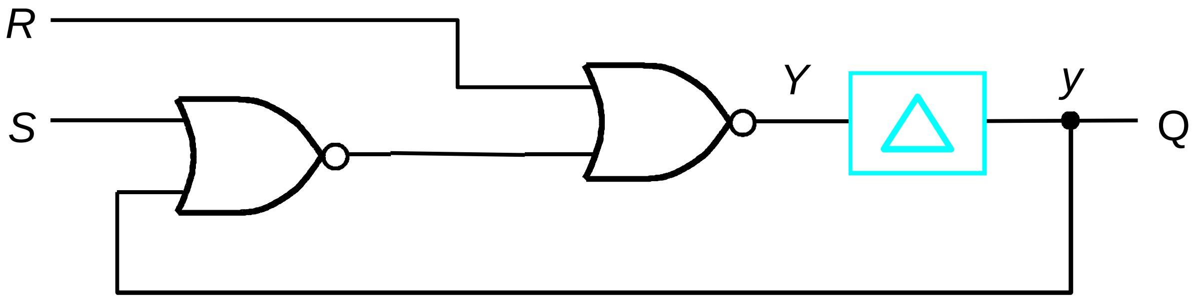
Analysis of a sequential asynchronous circuit

- By using a delay block, we can treat
  - y as the current state
  - Y as the next state



State table

- Thus, we can produce a state table where the next state Y depends on the inputs and the current state y



$$Y = \overline{R + \overline{(S + y)}}$$

State table

From statefunction to truth table

y	S	R	$Y=\overline{R+\overline{(S+y)}}$
0	0	0	$0=\overline{0+\overline{(0+0)}}$
0	0	1	$0=\overline{1+\overline{(0+0)}}$
0	1	0	$1=\overline{1+\overline{(1+0)}}$
0	1	1	$0=\overline{1+\overline{(1+0)}}$
1	0	0	$1=\overline{0+\overline{(0+1)}}$
1	0	1	$0=\overline{1+\overline{(0+1)}}$
1	1	0	$1=\overline{0+\overline{(1+1)}}$
1	1	1	$0=\overline{1+\overline{(1+1)}}$

Note: BV uses this binary code

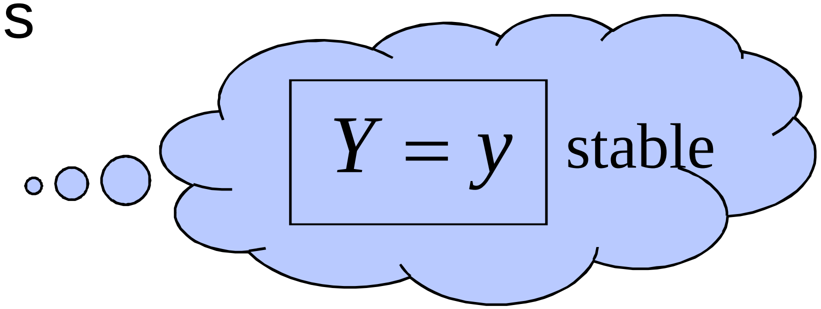
Present state y	Next state			
	SR = 00	01	10	11
	Y	Y	Y	Y
0	0	0	1	0
1	1	0	1	0

$$Y=\overline{R+\overline{(S+y)}}$$

Stable states

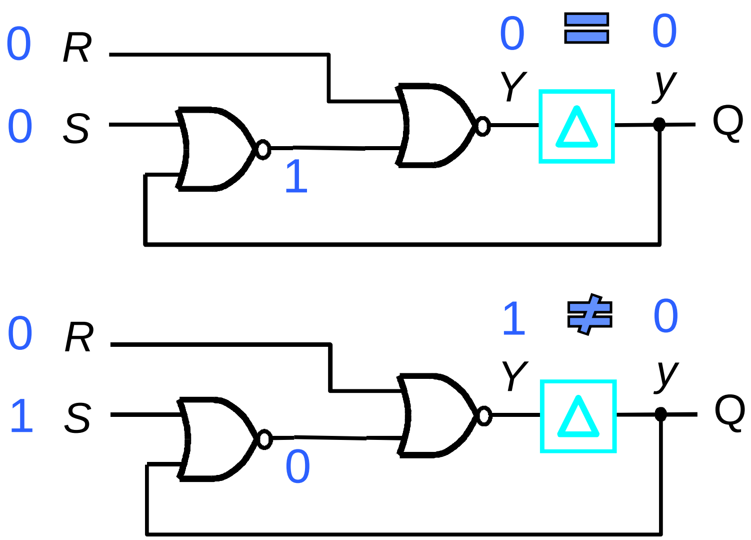
Present state y	Next state			
	SR = 00	01	10	11
	Y	Y	Y	Y
0	0	0	1	0
1	1	0	1	0

- Since we do not have flip-flops, but only combinational circuits, a state change can cause additional state changes
- A state is
  - stable if Y(t) = y(t + Δ)
  - unstable if Y(t) ≠ y(t + Δ)



Excitation table

- Stable states (next state = present state) are circled



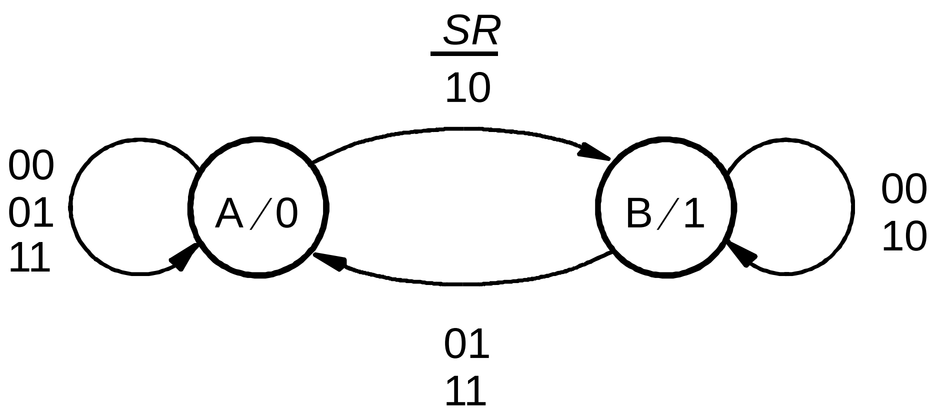
Present state y	Next state Y, $SR = 00$	01	10	11
0	0	0	1	0
1	1	0	1	0

Terminology

- When dealing with asynchronous sequential circuits, a different terminology is used
  - The state table called ***flow table***
  - The state-assigned state table is called ***excitation table***

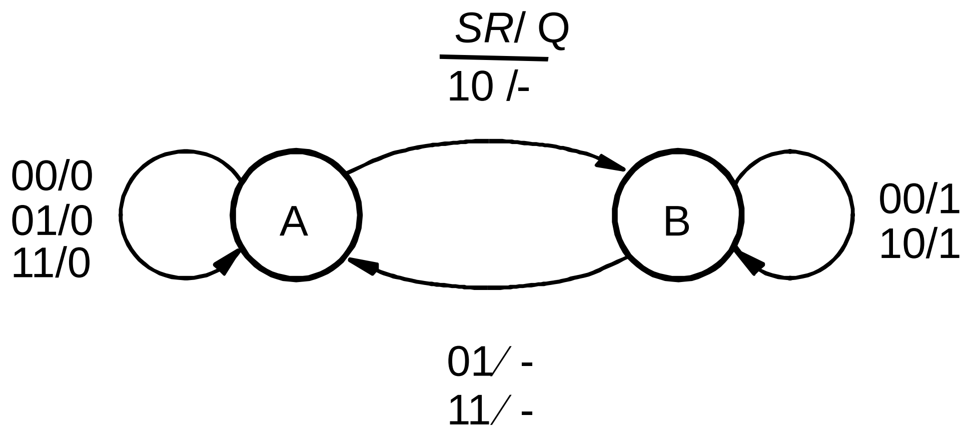
Flow table (Moore)

Present state	Next state				Output Q
	SR = 00	01	10	11	
A	A	A	B	A	0
B	B	A	B	A	1



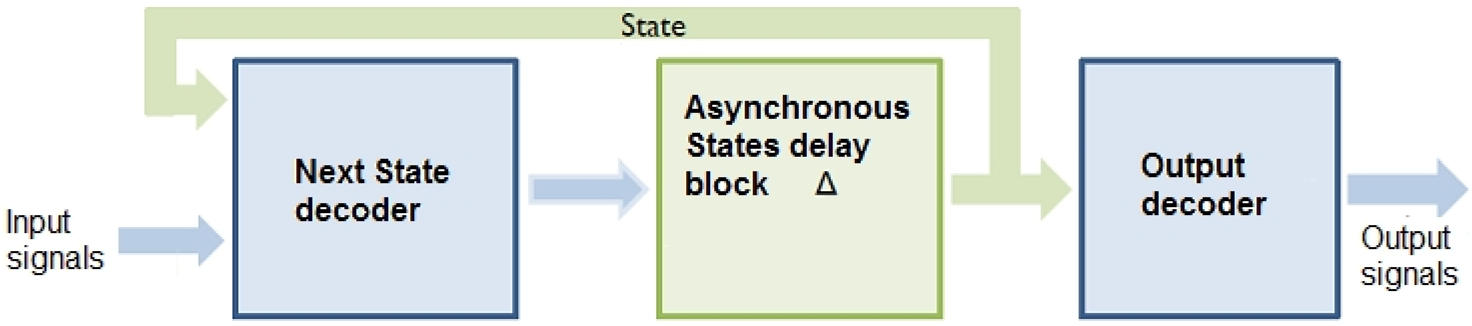
Flow Table (Mealy)

Present state	Next state				Output, Q			
	SR = 00	01	10	11	00	01	10	11
A	A	A	B	A	0	0	-	0
B	B	A	B	A	1	-	1	-

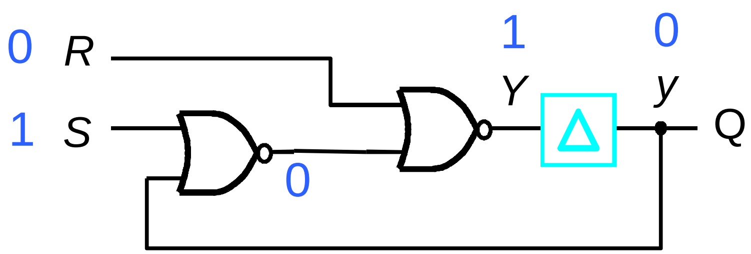


Do not care ('-') has been chosen for output decoder since output changes directly after the state transition (basic implementation)

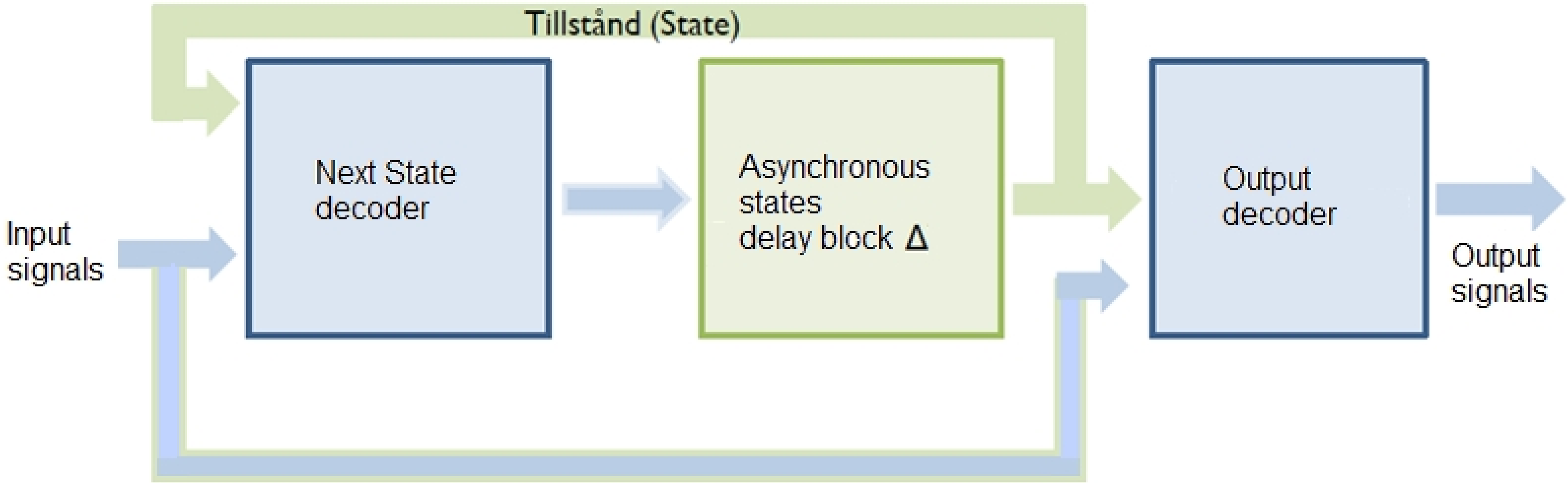
Asynchronous Moore compatible



- Asynchronous sequential circuits have similar structure as synchronous sequential circuits
- Instead of flip-flops one have a "delay block"



Asynchronous Mealy compatible

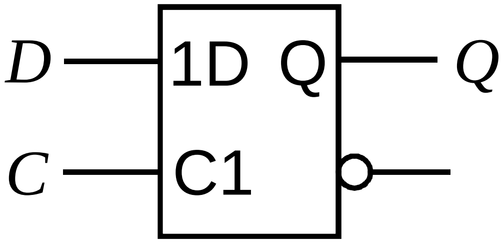


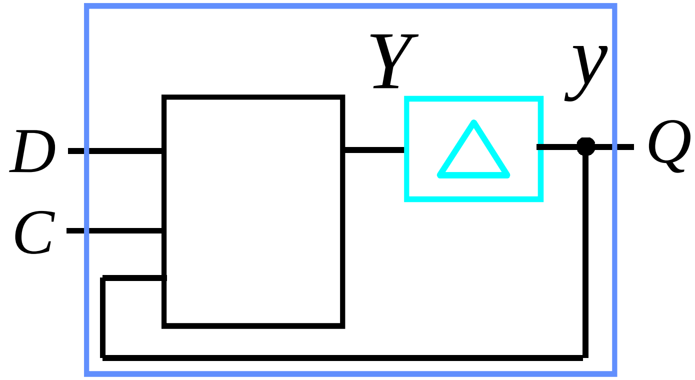
- Asynchronous sequential circuits have similar structure as synchronous sequential circuits
- Instead of flip-flops one have a "delay block"

Analysis of Asynchronous Circuits

- The analysis is done using the following steps:
  1) **Replace the feedbacks in the circuit with a delay element Δ.** The input of the delay element represents the next state Y while the output y represents the current state.
  2) Find out the **next-state and output expressions**
  3) Set up the corresponding **excitation table**
  4) Create a **flow table** and replace the encoded states with symbolic states
  5) Draw a **state diagram** if necessary

D-latch state function

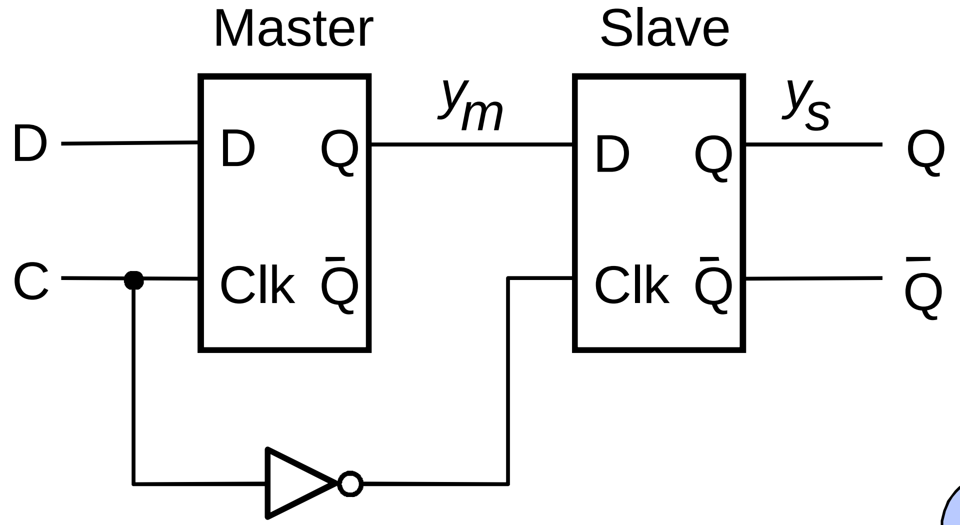




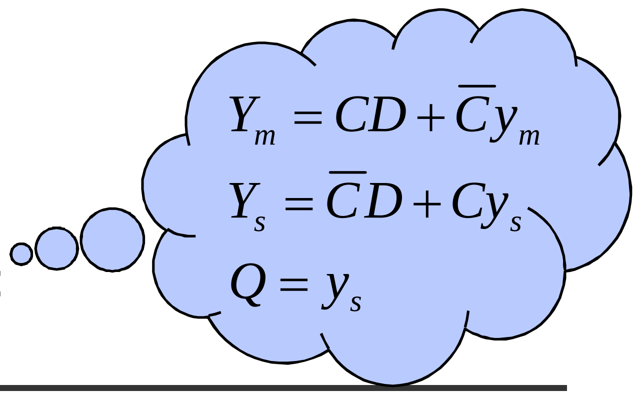
$$Y = D \cdot C + y \cdot \overline{C}$$

Example
Master-slave D flip-flop

- Master-slave D flip-flop is designed using two D-latches



The equations for the next state:



$$Y_m = CD + \overline{C}y_m$$

$$Y_s = \overline{C}D + Cy_s$$

$$Q = y_s$$

Excitation table

- From these equations, we can directly deduce excitation table

$$Y_m = CD + \overline{C}y_m$$

$$Y_s = \overline{C}D + Cy_s$$

$$Q = y_s$$

Excitation table

Present state $y_m\, y_s$	Next state $Y_m\, Y_s$				Output Q
	CD = 00	01	10	11	
00	00	00	00	10	0
01	00	00	01	11	1
10	11	11	00	10	0
11	11	11	01	11	1

Flow table

- We define four states S1, S2, S3, S4 and get the following flow table

Present state $y_m y_s$	Next state CD = 00	01	10	11	Output Q
	$Y_m Y_s$				
00	00	00	00	10	0
01	00	00	01	11	1
10	11	11	00	10	0
11	11	11	01	11	1

Excitation table

Present state	Next state CD = 00	01	10	11	Output Q
S1	S1	S1	S1	S3	0
S2	S1	S1	S2	S4	1
S3	S4	S4	S1	S3	0
S4	S4	S4	S2	S4	1

Flow table

Flow table

Present state	Next state CD = 00	01	10	11	Output Q
S1	S1	S1	S1	S3	0
S2	S1	S1	S2	S4	1
S3	S4	S4	S1	S3	0
S4	S4	S4	S2	S4	1

- Remember: Only one input can be changed simultaneously
- Thus, some transitions never occur!

Flow table (Impossible transitions)

Present state	Next state CD = 00	01	10	11	Output Q
S1	S1	S1	S1	S3	0
S2	S1	S1	S2	S4	1
S3	~~S4~~	S4	S1	S3	0
S4	S4	S4	S2	S4	1

- State S3
  - The only stable state is S3 with input combination 11
  - Only one input can be changed => possible transitions are (11 => 01, 11 => 10)
    - These transitions originate in S3!
    - The input combination 00 in S3 is not possible!
    - The input combination 00 is set to don't care!

Flow table
(Impossible transitions)

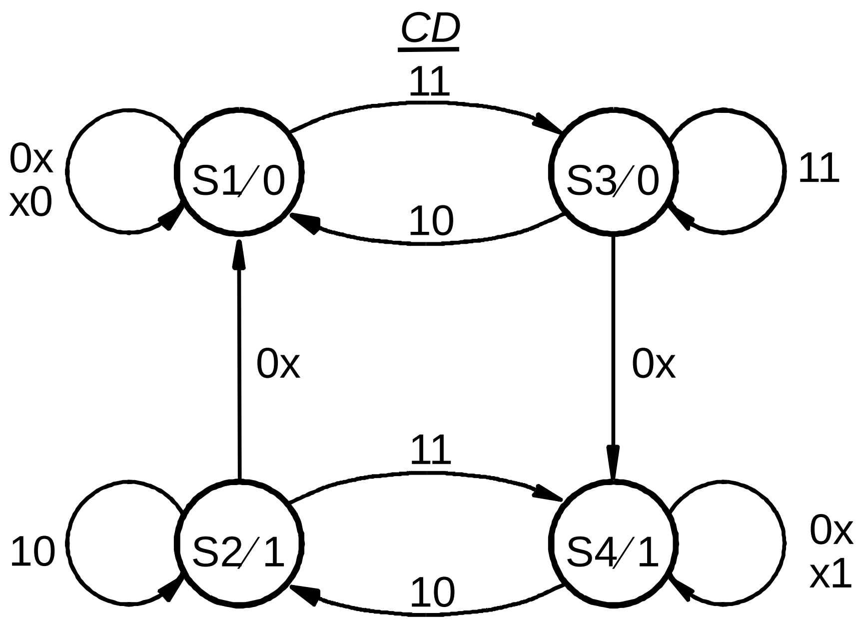
Present state	Next state CD = 00	01	10	11	Output Q
S1	S1	S1	S1	S3	0
S2	S1	~~S1~~	S2	S4	1
S3	-	S4	S1	S3	0
S4	S4	S4	S2	S4	1

- State S2
  - The only stable state is S2 with input combination 10
  - Only one entry can be changed => possible transitions are (10 => 11, 10 => 00)
    - These transitions originate in S2!
    - The input combination 01 in S2 is not possible!
    - The input combination 01 is set to don't care!

State Diagram
Master-slave D flip-flop

Present state	Next state CD = 00	01	10	11	Output Q
S1	S1	S1	S1	S3	0
S2	S1	-	S2	S4	1
S3	-	S4	S1	S3	0
S4	S4	S4	S2	S4	1

Flow table



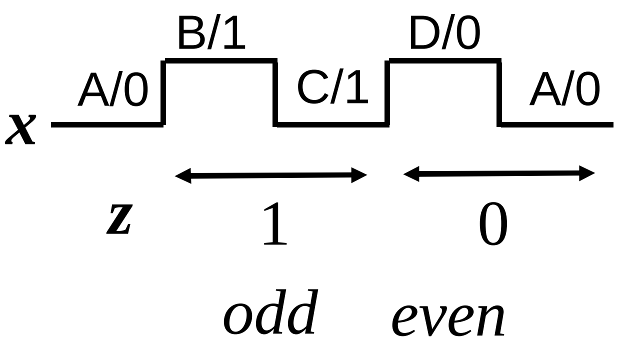
State diagram

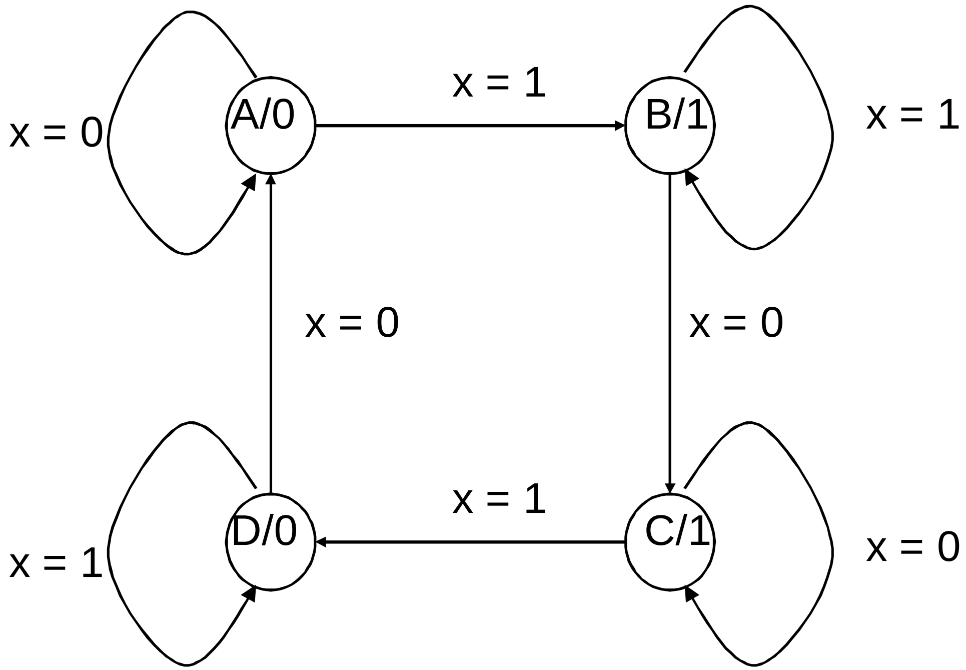
Synthesis of asynchronous circuits

- The synthesis is carried out using the following steps:
  1) Create a **state diagram** according to the functional description
  2) Create a **flow table** and **reduce the number of states** if possible
  3) Assign codes to the states and create **excitations table**
  4) **Determine expressions** (transfer functions) for the next state and outputs
  5) **Construct a circuit** that implements the above expressions

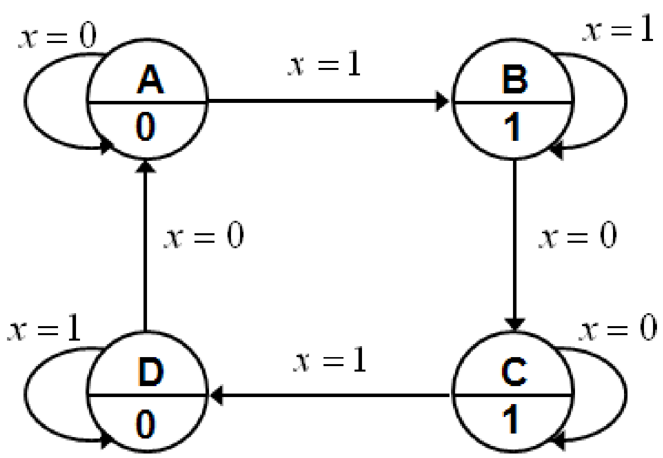
Example: Serial Parity Generator
Step 1: Create a state diagram

- Input x
- Output z
- z = 1 if the number of pulses applied to x is odd
- z = 0 if the number of pulses applied to x is even



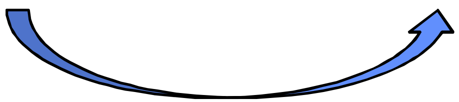


Step 2: Flow table



Pres state	Next State		z
	x=0	x=1	
A	A	B	0
B	C	B	1
C	C	D	1
D	A	D	0

State diagram



Flow table

Step 3: Assign state codes

Pres state	Next State x=0	Next State x=1	z
A	A	B	0
B	C	B	1
C	C	D	1
D	A	D	0

Flow table

Pres state y_2y_1	Next State x=0 Y_2Y_1	Next State x=1 Y_2Y_1	z
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0

Excitation table

A:00, B:01, C:10, D:11 - binary code?
Which encoding is good?

Step 3: Assign state codes
Which encoding is good?

Assume → A:00, B:01, C:10, D:11

Pres state y_2y_1	Next State X=0 ← 1 Y_2Y_1		Q
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0

Bad encoding (HD=2!)

Suppose

$X = 1 \;\; Y_2Y_1 = 11$

Then

$X \rightarrow 0 \rightarrow Y_2Y_1 = 00?$

$11 \rightarrow 10!$

$11 \rightarrow 01 \rightarrow 10! \quad ? \rightarrow 00$

We will never reach 00?

Step 3: Assign state codes
Which encoding is good?

A:00, B:01, C:10, D:11

Pres state	Next State		z
	x=0	x=1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0

Poor encoding (HD = 2)
If we are in 11 under input w = 1 and input change to w = 0, the circuit should change to 00

A:00, B:01, C:11, D:10

Pres state	Next State		z
	x=0	x=1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

Good encoding (HD = 1)

Step 4:
Draw Karnaugh maps

Pres state	Next State		z
	x=0	x=1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

$x \backslash y_2y_1$	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y_2 = \overline{x}y_1 + y_2y_1 + xy_2$

$x \backslash y_2y_1$	00	01	11	10
0	0	1	1	0
1	1	1	0	0

$Y_1 = x\overline{y_2} + \overline{y_2}y_1 + \overline{x}y_1$

$y_2 \backslash y_1$	0	1
0	0	1
1	0	1

$z = y_1$

They red circles are needed to avoid hazards (see later Section)!

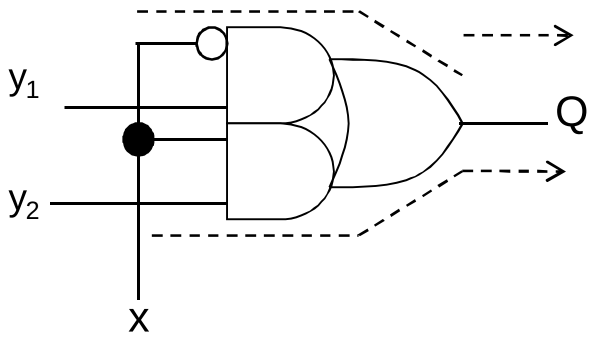
What is a Hazard?

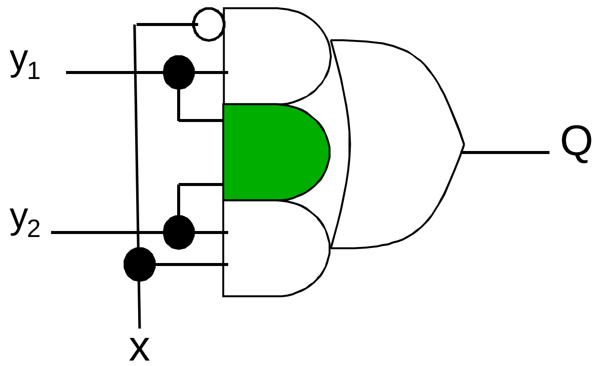
- Hazard is a term that means that there is a danger that the output value is not stable, but it can have glitches at certain input combinations
- Hazard occurs when paths from different inputs to the output have different lengths
- To avoid this, we must add implicants to cover the ”dangerous” transitions

Examples of hazard: MUX

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$Y_2 = \overline{x}\, y_1 + y_2 y_1 + x\, y_2$





During the transition from the $(xy_2y_1) = (111)$ to (011), the output Q has a glitch, as the path from x to Q is longer through the upper AND gate than through the lower AND gate (racing).

MORE ABOUT hazard in the next lecture!

Step 5: Complete circuit

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	0	0	1	1

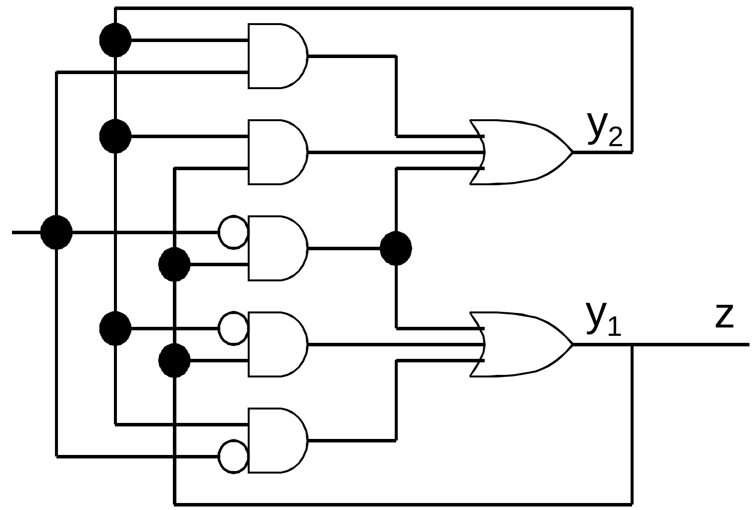
$Y_2 = \bar{x}\, y_1 + y_2 y_1 + x\, y_2$

y_2 \ y_1	0	1
0	0	1
1	0	1

$z = y_1$

x \ y_2y_1	00	01	11	10
0	0	1	1	0
1	1	1	0	0

$Y_1 = x\bar{y}_2 + \bar{y}_2 y_1 + \bar{x} y_1$



More on state encoding

- In asynchronous sequential circuits, it is impossible to guarantee that the two state variables change value simultaneously
  - Thus, a transition 00 => 11 results in
    - a transition 00 => 01 => ???
    - a transition 00 => 10 => ???
- To ensure correct operation, all state transitions **MUST** have Hamming distance 1
  - The Hamming distance is the number of bits in which two binary numbers differ
    - Hamming distance between 00 and 11 is 2
    - Hamming distance between 00 and 01 is 1

State encoding

- Procedure to obtain good codes:

  1) Draw the transition diagram along the edges of the hypercube defined by the codes

  2) Remove any crossing lines by

     a) swapping two adjacent nodes

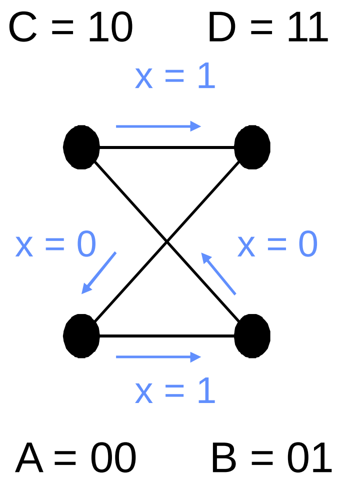
     b) exploiting available unused states

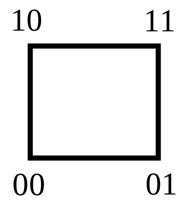
     c) introducing more dimensions in the hypercube

State encoding
Example: Serial Parity Generator

Pres state	Next State		z
	x=0	x=1	
y_2y_1	Y_2Y_1		
00	00	01	0
01	10	01	1
10	10	11	1
11	00	11	0

A:00, B:01, C:10, D:11



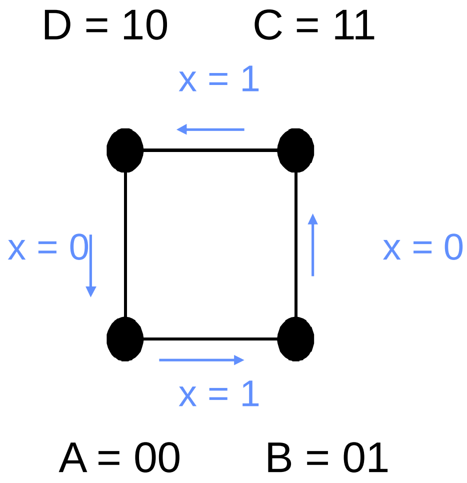


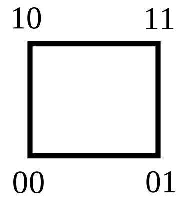
Poor coding -
Hamming Distance = 2
(Intersecting lines)

State encoding
Example: Serial Parity Generator

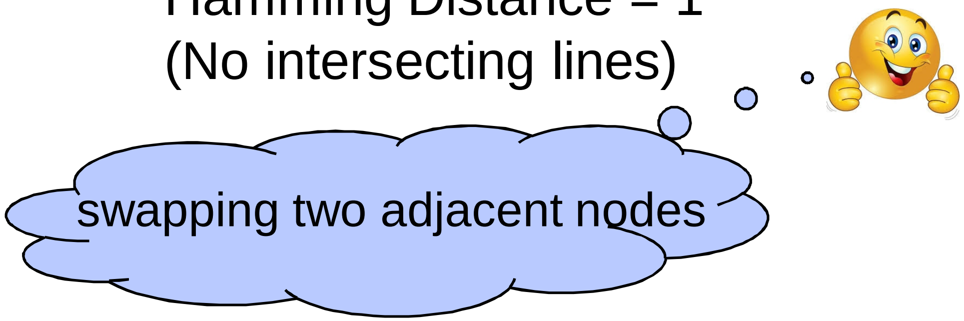
Pres state y_2y_1	Next State x=0 Y_2Y_1	Next State x=1 Y_2Y_1	z
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

A:00, B:01, C:11, D:10

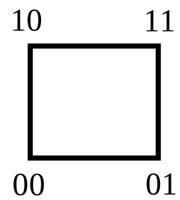




Good coding
Hamming Distance = 1
(No intersecting lines)



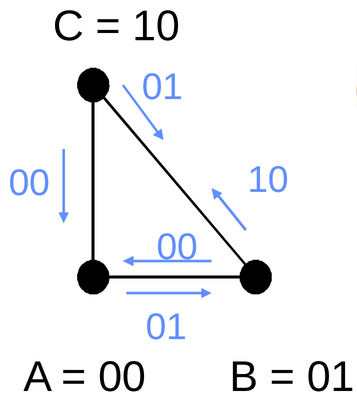
State encoding
Exploiting unused states



Flow table from Fig. 9.21a of BV textbook

Note: BV uses this binary code

Present state	Next state r_2r_1 = 00	01	10	11	Output g_2g_1
A	A	B	C	-	00
B	A	B	C	B	01
C	A	B	C	C	10

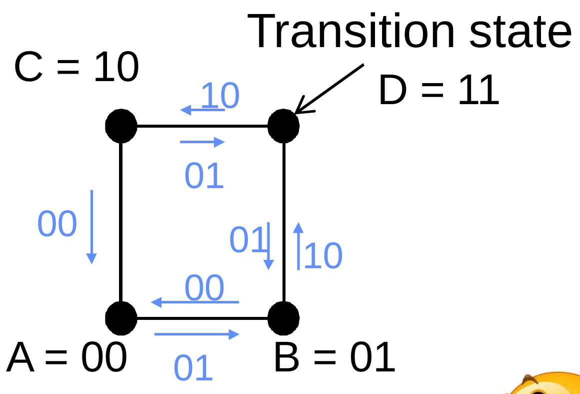


Poor coding

In the transition from B to C (or C to B) has the Hamming distance 2!
Danger to get stuck in an unspecified state (with code 11)!

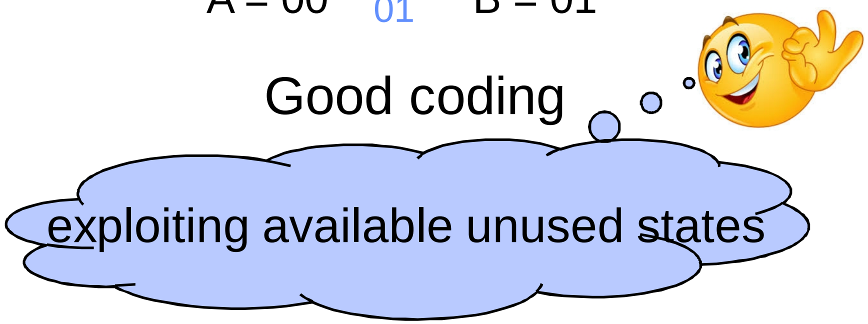
State encoding
Exploiting aunused states

- Solution: Introduce a transition state that ensures that you do not end up in an unspecified state!



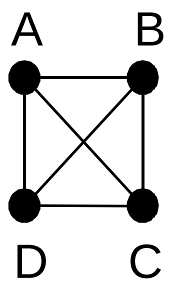
Present state	Nextstate r_2r_1 = 00	01	11	10	Output g_2g_1
A	A	B	-	C	00
B	A	B	B	C	01
C	A	B	C	C	10

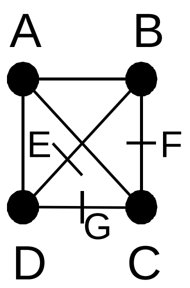
	Present state y_2y_1	Nextstate Y_2Y_1, r_2r_1 = 00	01	11	10	Output g_2g_1
A	00	00	01	-	10	00
B	01	00	01	01	11	01
D	11	-	01	-	10	dd
C	10	00	11	10	10	10

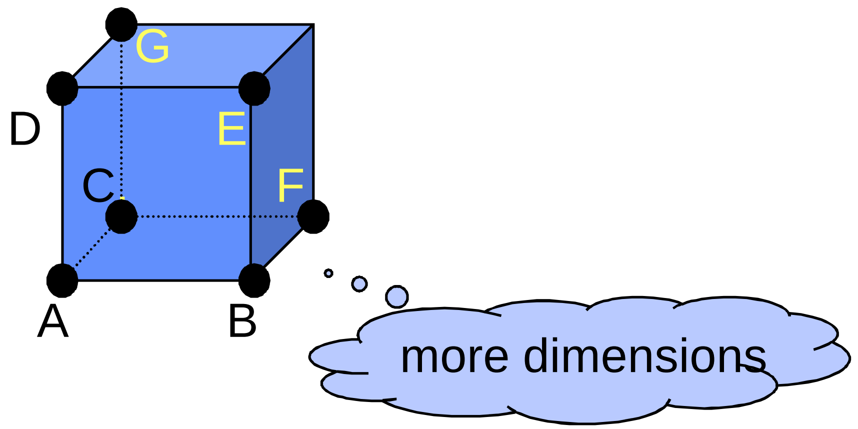


State encoding: Additional states (more dimensions)

- One can increase the number of dimensions in order to implement stable state transitions



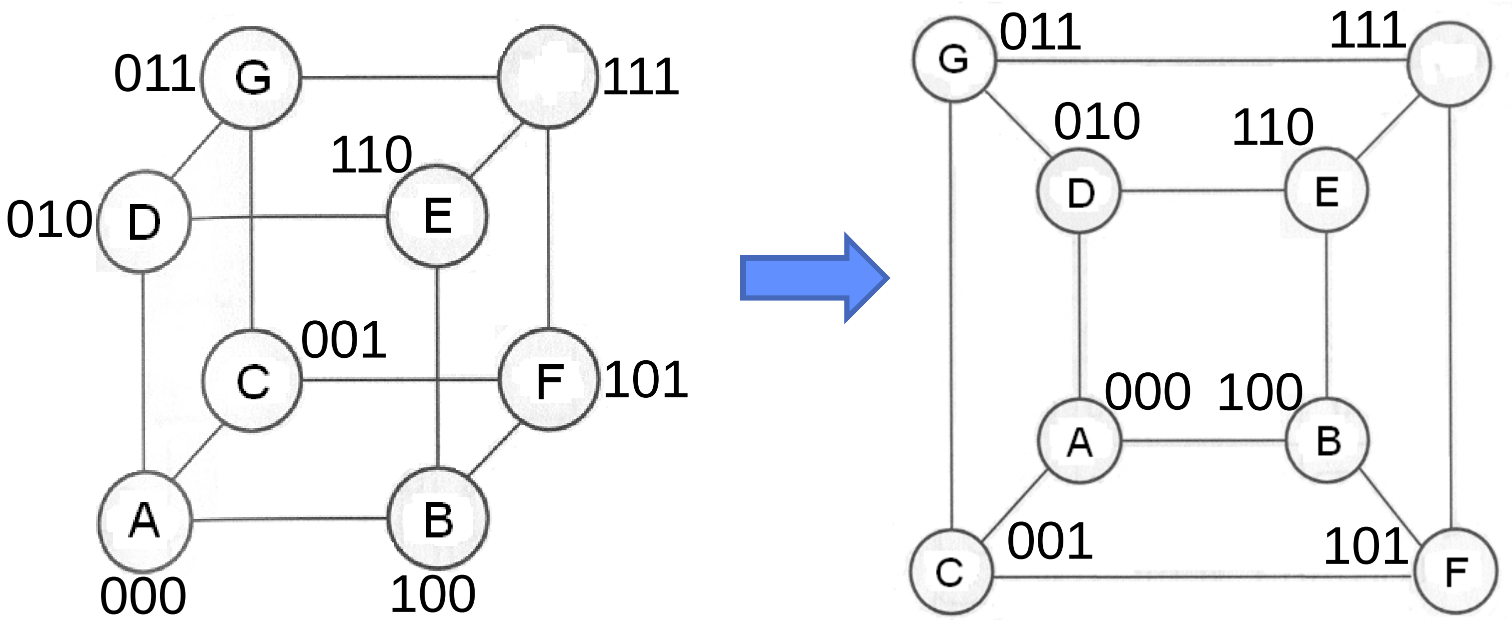




If it is not possible to redraw a diagram for HD = 1, we can add more states by adding extra dimensions. We take the nearest largest hypercube and draw the transitions through the available non steady states.

State encoding: Additional states (more dimensions)

- It's easier to draw a "flat" 3D cube (perspective, is then from the front)



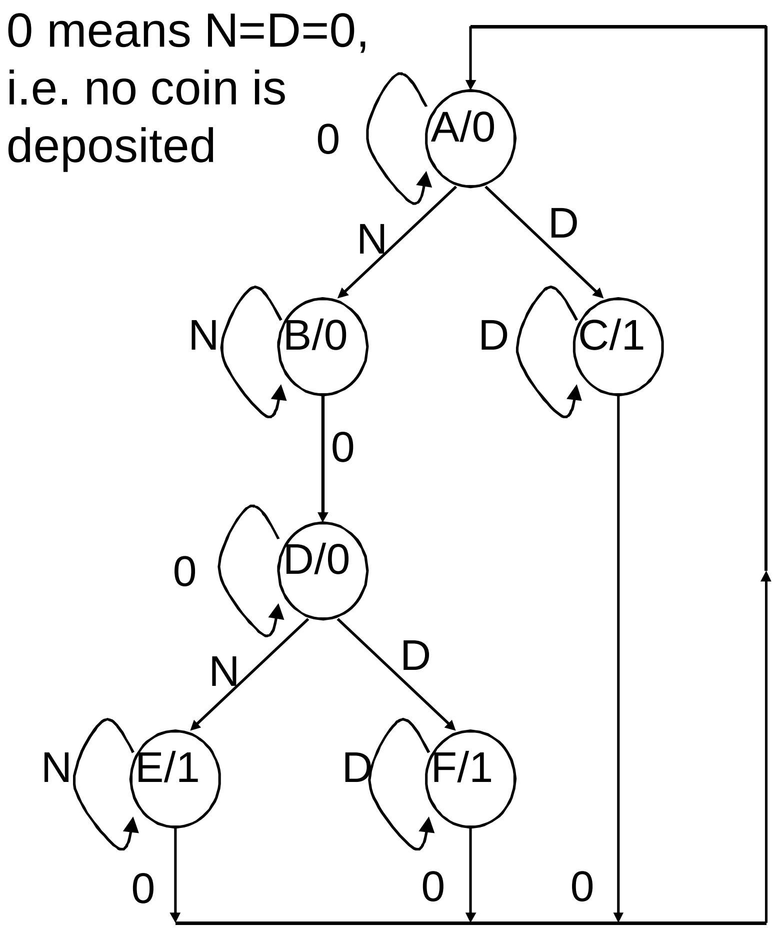
State minimization

- Procedure for minimizing the number of states
  1. Form equivalence classes
  2. Minimize equivalence classes (state reduction)
  3. Form state diagrams either for Mealy or Moore
  4. Merge compatible states in classes. Minimize the number of classes simultaneously. Each state can only belong to one classes
  5. Construct the reduced flow table by merging rows in the selected classes
  6. Repeat steps 3-5 to see if more minimizations can be done

Example
Candy Machine (BV page 610)

- Candy machine has two inputs:
  – N: nickel (5 cents)
  – D: dime (10 cents)
- A candy bar costs 10 cents
- The machine will not return any change if there is 15 cents in the candy machine (a candy bar returned)
- The output z is active if there is enough money for a piece of candy

State Diagram and Flow Chart



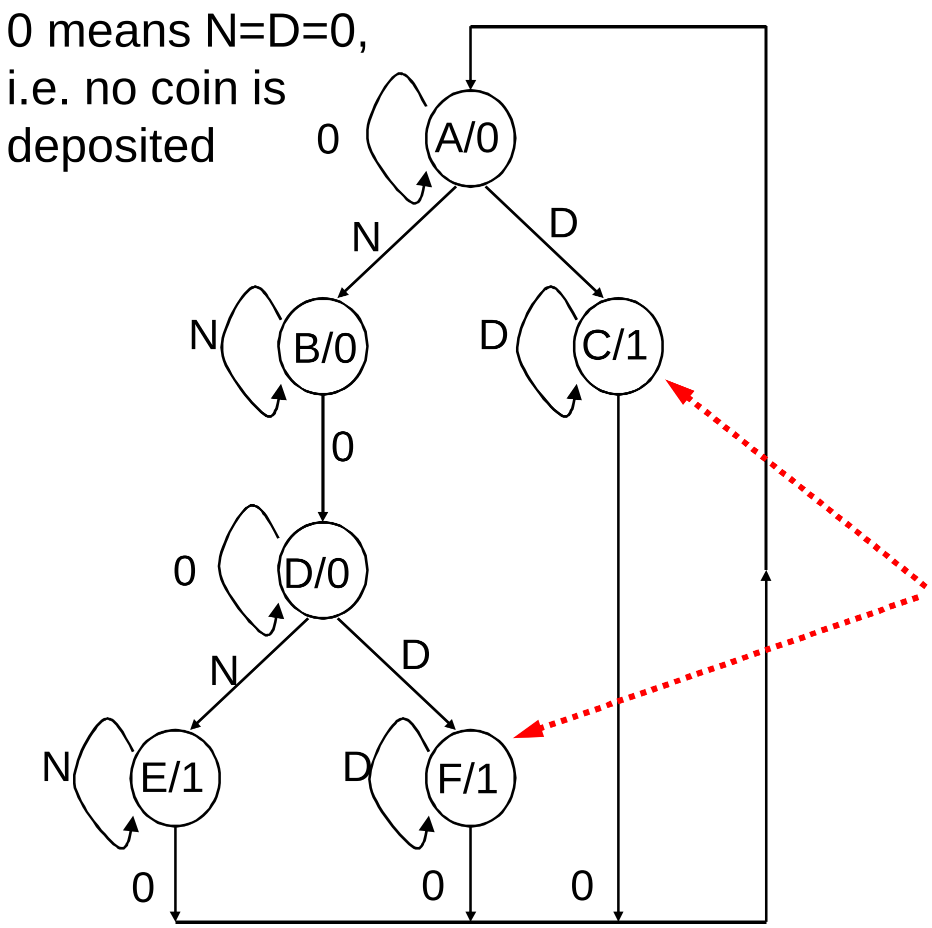
- *You can't insert two coins at the same time!*

Pres state	Next State X=00	01	10	11	z
A	A	B	C	-	0
B	D	B	-	-	0
C	A	-	C	-	1
D	D	E	F	-	0
E	A	E	-	-	1
F	A	-	F	-	1

(X = DN)

A flow table that contains only one stable state per row is called *primitive* flow table.

State Diagram and Flow Chart



State Minimization means that two states may be equivalent, and if so, replaced by one state to simplify the state diagram, and network.

One can easily see that state C and F could be replaced by one state, as a candy always be ejected after a Dime regardless of previous state.

Step 1: Form and minimize equivalence classes

1. Forming equivalence classes. To be in the same class, the following should hold for states:
   – Outputs must have the same value
   – Stable states must be at the same positions
   – Don’t cares for next state must be in the same positions
2. Minimize equivalence classes (state-reduction)

State reduction

- Outputs must have the same value

  P_1 = (ABD)(CEF)

- Stable states must be at the same positions

  P_2 = (AD)(B)(CF)(E)

- Don't cares for next state must be in the same positions

  P_2 = (AD)(B)(CF)(E)

Primitive flow table

Pres state	Next State X=00	01	10	11	Q
A	A	B	C	-	0
B	D	B	-	-	0
C	A	-	C	-	1
D	D	E	F	-	0
E	A	E	-	-	1
F	A	-	F	-	1

State reduction

- Successors must be in the same class

$C,F_{00} \rightarrow (AD), (AD)$
$C,F_{01} \rightarrow -, -$
$C,F_{10} \rightarrow (CF), (CF)$
$C,F_{11} \rightarrow -, -$

$A,D_{00} \rightarrow (AD), (AD)$
$A,D_{01} \rightarrow (B),(E)$
$A,D_{10} \rightarrow (CF), (CF)$
$A,D_{11} \rightarrow -, -$

$P_2 = (AD)(B)(CF)(E)$
$P_3 = (A)(D)(B)(CF)(E)$
$P_3 = P_4$

Primitive flow table

Pres state	Next State X=00	01	10	11	Q
A	A	B	C	-	0
B	D	B	-	-	0
C	A	-	C	-	1
D	D	E	F	-	0
E	A	E	-	-	1
F	A	-	F	-	1

Resulting flow table

Pres state	Next State X=00	01	10	11	Q
A	A	B	C	-	0
B	D	B	-	-	0
C	A	-	C	-	1
D	D	E	C	-	0
E	A	E	-	-	1

Step 2:
Merging states

3. Construct state diagram either for Mealy or Moore
4. Merge **compatible states** in groups. Minimize the number of groups simultaneously. Each state may belong to one group only.
5. Construct the reduced flow table by merging rows in the selected groups
6. Repeat steps 3-5 to see if more minimizations can be done

Merging states

- Two states are compatible and can be merged if the following applies
  1. at least one of the following conditions apply to all input combinations
     - both S_i and S_j have the same successor, or
     - both S_i and S_j are stable, or
     - the successor of S_i or S_j, or both, is unspecified
  2. For a Moore machine, in addition the following should hold
     - both S_i and S_j have the same output values whenever specified (not necessary for a Mealy machine)

Merging states

Resulting flow table

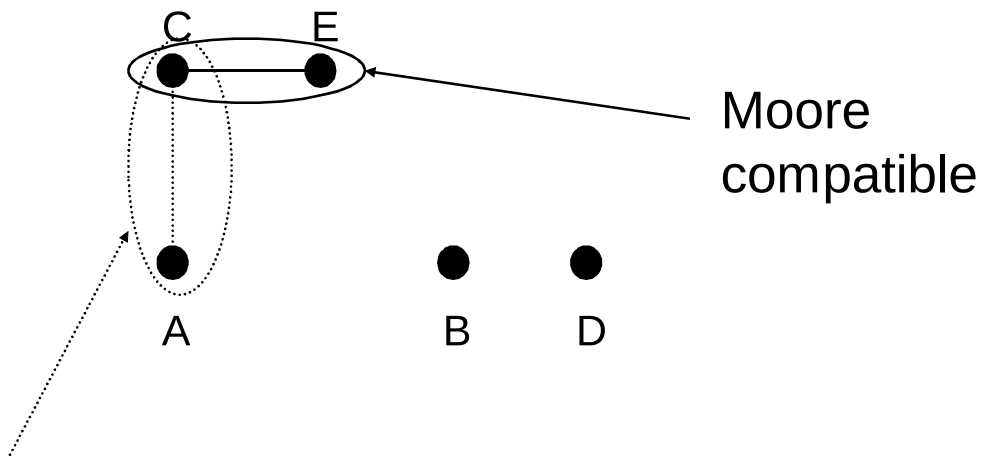
Pres state	Next State X=00	01	10	11	Q
A	A	B	C	-	0
B	D	B	-	-	0
C	A	-	C	-	1
D	D	E	C	-	0
E	A	E	-	-	1

Each row will be a point in a compatibility graph

- both S_i and S_j have the same successor, or
- both S_i and S_j are stable, or
- the successor of S_i or S_j, or both, is unspecified

Moreover, both S_i and S_j must have the same output whenever specified

Compatibility graph



Mealy-compatible: In state A (X = 00) the output is 0, in state C the output is 1

An illustrative example

Equivalence classes

P_1= (AG) (BL) (C) (D) (E) (F) (HK) (J)
P_2= (A) (G) (BL) (C) (D) (E) (F) (HK) (J)
P_3= P_2

Primitive flow table

Pres state	Next State X=00	01	10	11	Q
A	A	F	C	-	0
B	A	B	-	H	1
C	G	-	C	D	0
D	-	F	-	D	1
E	G	-	E	D	1
F	-	F	-	K	0
G	G	B	J	-	0
H	-	L	E	H	1
J	G	-	J	-	0
K	-	B	E	K	1
L	A	L	-	K	1

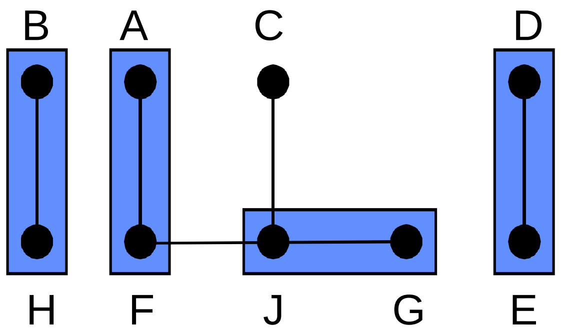
Reduced flow table

Pres state	Next State X=00	01	10	11	Q
A	A	F	C	-	0
B	A	B	-	H	1
C	G	-	C	D	0
D	-	F	-	D	1
E	G	-	E	D	1
F	-	F	-	H	0
G	G	B	J	-	0
H	-	B	E	H	1
J	G	-	J	-	0

An illustrative example (cont'd)

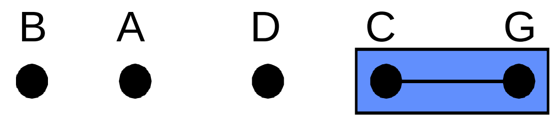
Reduced flow table

Pres state	Next State X=00	01	10	11	Q
A	A	F	C	-	0
B	A	B	-	H	1
C	G	-	C	D	0
D	-	F	-	D	1
E	G	-	E	D	1
F	-	F	-	H	0
G	G	B	J	-	0
H	-	B	E	H	1
J	G	-	J	-	0



Pres state	Next State X=00	01	10	11	Q
A	A	A	C	B	0
B	A	B	D	B	1
C	G	-	C	D	0
D	G	A	D	D	1
G	G	B	G	-	0

An illustrative example (cont'd)



Reduced flow table

Pres state	Next State X=00	01	10	11	Q
A	A	A	C	B	0
B	A	B	D	B	1
C	G	-	C	D	0
D	G	A	D	D	1
G	G	B	G	-	0

Final flow table

Pres state	Next State X=00	01	10	11	Q
A	A	A	C	B	0
B	A	B	D	B	1
C	C	B	C	D	0
D	C	A	D	D	1

Summary

- Asynchronous state machines
  - Based on analysis of combinational circuits with feedback
  - All flip-flops and latches are asynchronous state machines
- A similar theory as for synchronous state machines can be applied
  - Only one input or state variable can be changed at a time!
  - We must also take into account the problem with hazards
- Next lecture: BV pp. 640-648, 723-724