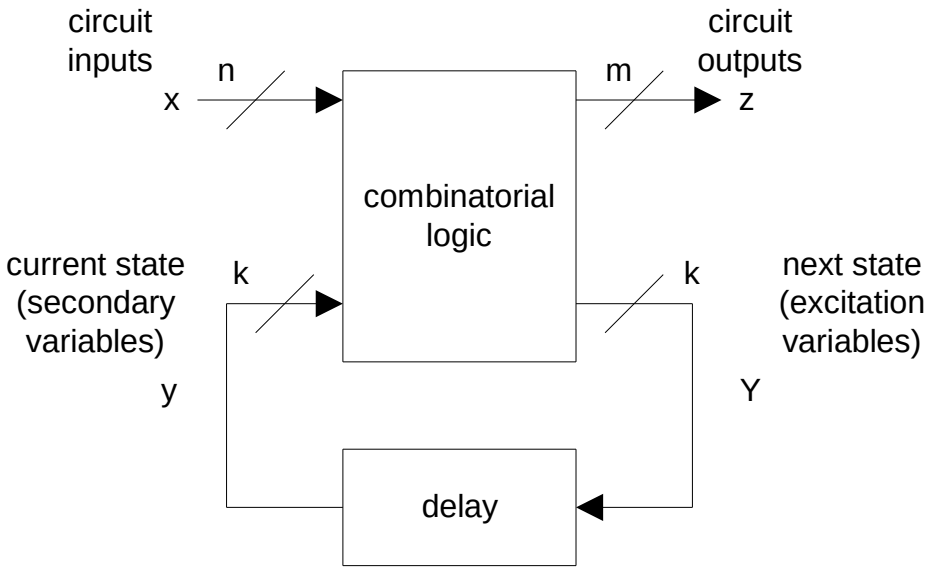
Asynchronous Sequential Circuits

- Type of circuit without clocks, but with the concept of memory.
- Concept of memory is obtained via un-clocked latches and/or circuit delay.
- Changes in input variables cause changes in states.
- Asynchronous sequential circuits resemble combinatorial circuits with feedback paths.
- The design of synchronous circuits is more difficult than synchronous circuits using flip-flops and clocks.

Block Diagram

- General block diagram:



- Note difference in “little y” secondary variables and “capital Y” excitation variables.
- Delay elements are hypothetical, and typically are a result of gate delays.
- Note: When inputs change, excitation variables Y change. It takes additional delay for the secondary variables (current state) to assume the values of the excitation variables (next state).

Definition – Stability and Fundamental Mode Operation

- Stability:
  - For a given set of inputs (i.e., values), the system is **STABLE** if the circuit eventually reaches **steady state** and the excitation variables and secondary variables are equal and unchanging (little y = capital Y), otherwise the circuit is **UNSTABLE**.

- Fundamental Mode:
  - A circuit is operating in fundamental mode if we assume/force the following restrictions on how the inputs can change:
    - Only ONE input is allowed to change at a time, AND
    - The input changes only after the circuit is STABLE.

Textbook

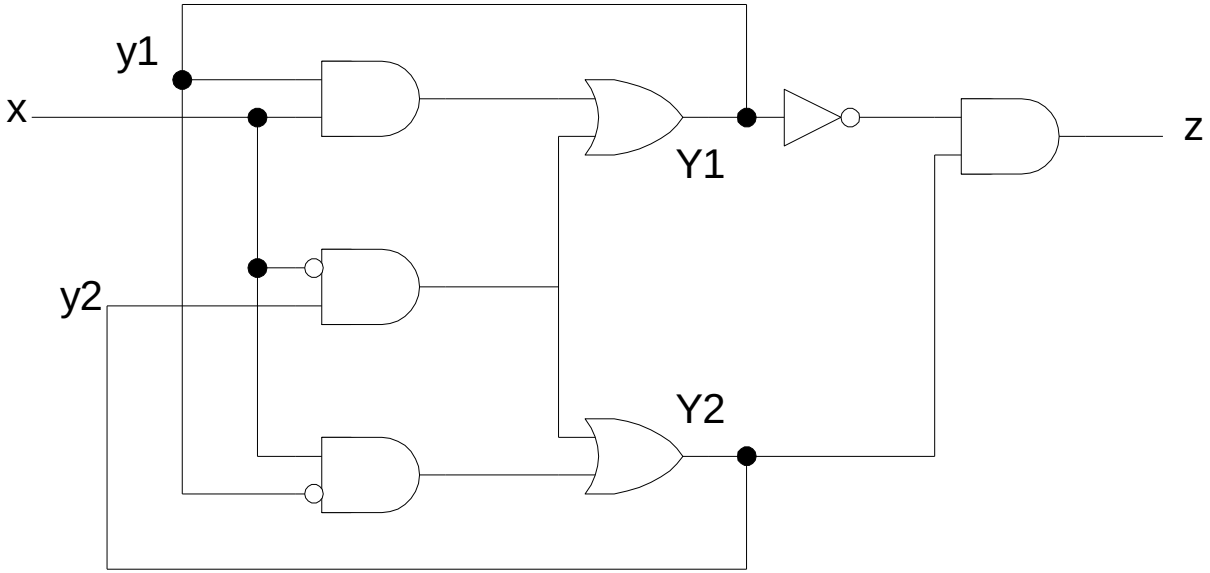
- Introduction to asynchronous circuits is covered in Chapter 9, Section 9.1 of the course textbook.

Asynchronous Circuit Analysis

- Asynchronous circuits are identified by:
  - The presence of combinatorial feedback paths, and/or
  - The presence of un-clocked storage elements (i.e., latches).
- Analysis involves obtaining a table or diagram that describes the sequence of internal states and outputs as a function of changes in the circuit inputs.
- The tables we will try to obtain are **transition tables** and **flow tables**.

Example

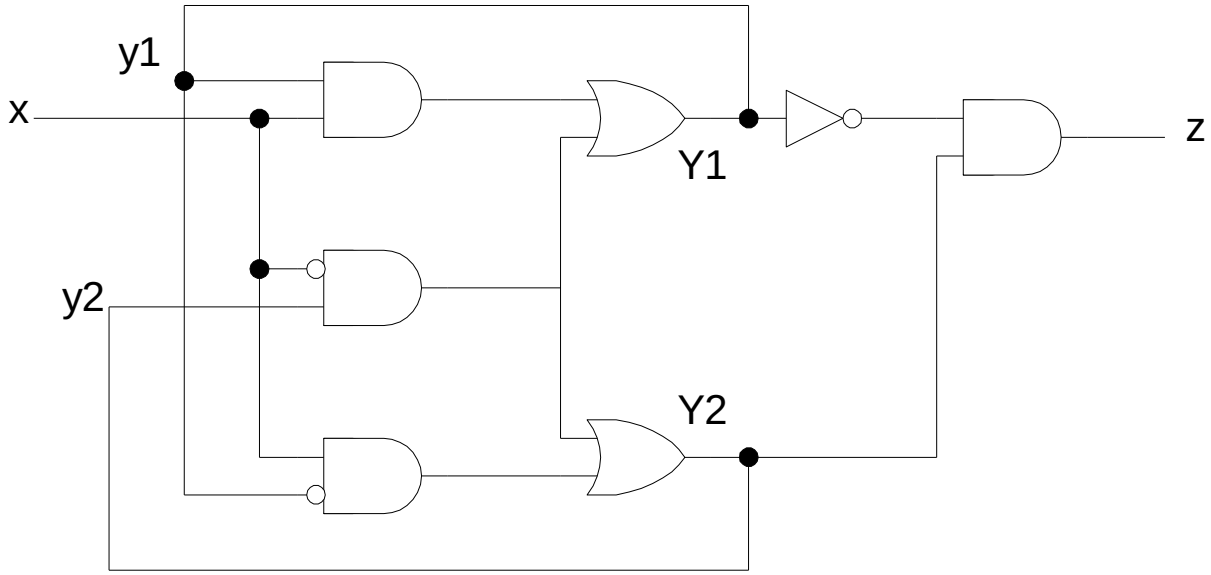
- Consider the following circuit that has combinatorial feedback paths (and is therefore identified as asynchronous). No apparent latches in the circuit:



- Circuit has one input (x), one output (z), two secondary variables (y1, y2) and two excitation variables (Y1, Y2).

Analysis Example - Equations

- Write logic equations for the excitation variables in terms of the circuit inputs and secondary variables:



$$Y_1 = xy_1 + \bar{x}y_2$$

$$Y_2 = \bar{x}y_2 + x\bar{y}_1$$

- Write logic equations for circuit outputs in terms of the circuit inputs and secondary variables:

$$z = \bar{y}_1 y_2$$

Analysis Example – Transition Table

- Using these equations, we can write a **transition table** that shows excitation variables and outputs as a function of inputs and secondary variables:

$$Y_1 = xy_1 + \overline{x}y_2$$

$$Y_2 = \overline{x}y_2 + x\overline{y}_1$$

$$z = \overline{y}_1 y_2$$

curr state	next state		output	
	x=0	x=1	x=0	x=1
y2y1	Y2Y1	Y2Y1	z	z
00	(00)	10	0	0
01	00	(01)	0	0
10	11	(10)	1	1
11	(11)	01	0	0

- **Note that stable states (secondary variables equal to excitation variables) are circled.**

Analysis Example – Flow Table

- We can also create a **flow table**, which is just the transition table with binary numbers replaced with symbols (e.g., let **a = 00, b = 01, c = 10** and **d = 11**):

curr state	next state		output	
	x=0	x=1	x=0	x=1
y2y1	Y2Y1	Y2Y1	z	z
00	00	10	0	0
01	00	01	0	0
10	11	10	1	1
11	11	01	0	0

curr state	next state		output	
	x=0	x=1	x=0	x=1
y2y1	Y2Y1	Y2Y1	z	z
a	a	c	0	0
b	a	b	0	0
c	d	c	1	1
d	d	b	0	0

- We could proceed to draw something like a state diagram from this information, if we choose…

Analysis Example –Flow Table Alternative

- Another way to draw a flow table:

	x=0	x=1
a	a , 0	c , 0
b	a , 0	b , 0
c	d , 1	c , 1
d	d , 0	b , 0

- Left-most column shows current state (secondary variables), and the inputs are listed across the top.
- Entries in the matrix show the next state (excitation variables) and output values.

Primitive Flow Tables

- Flow table with **only one stable state per row is called a primitive flow table.**
- E.g., a primitive flow table:

	x = 0	x = 1
a	**a**	b
b	c	**b**
c	**c**	d
d	a	**d**

Primitive Flow Tables

- E.g., a flow table that is not a primitive flow table:

	x1x2			
	00	01	11	10
a	a	a	a	b
b	a	a	b	b

Analysis Summary

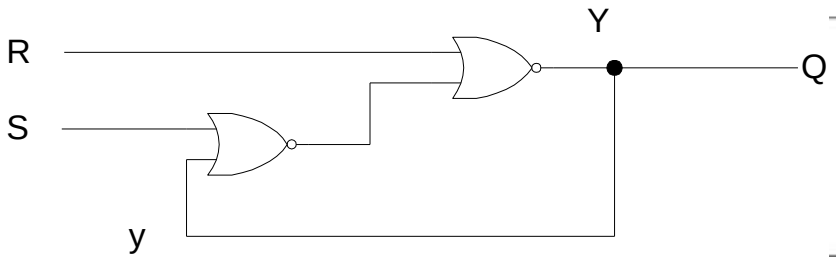
- Procedure to determine transition table and/or flow table from a circuit with combinatorial feedback paths:
  - Determine feedback paths.
  - Label Y (excitation variables) at output and y (secondary variables at input).
  - Derive logic expressions for Y (excitation variables) in terms of circuit inputs and secondary variables. Do the same for circuit outputs.
  - Create a transition table and flow table.
  - Circle stable states where Y (excitation variables) are equal to y (secondary variables).

Latch Analysis

- We can use the previous analysis technique to see how latches work…
- We will consider SR (built with NOR gates) and S'R' (built with NAND gates) Latches.

SR Latches

- We can analyze an SR latch using the previous technique:



S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	(after $S=1, R=0$)
0	1	0	1	
0	0	0	1	(after $S=0, R=1$)
1	1	0	0	

SR

y	00	01	11	10
0	**0**	**0**	**0**	**1**
1	1	0	0	1

Y=Q

- Equations derived for secondary variable (same equation for output):

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

$$Y' = R + (S+y)'$$

$$Y' = (R'.(S+y))'$$

$$\Rightarrow Y = R'(S+y) = R'S + R'y$$

- Since we want to avoid the SR=11 situation, we can write:

$$Y = S + \overline{R}y \quad \text{if} \quad SR = 0$$

SR Latches

y \ SR	00	01	11	10
0	0	0	0	1
1	1	0	0	1

Y=Q

- Can derive the transition table and the flow table:

Don't care

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

$$Y = S + \overline{R}y \quad \text{if} \quad SR = 0$$

curr state	next state SR=00	01	11	10	output
y	Y	Y	Y	Y	
0	0	0	0	1	0
1	1	0	0	1	1

curr state	next state SR=00	01	11	10	output
y	Y	Y	Y	Y	
a	a	a	a	b	0
b	b	a	a	b	1

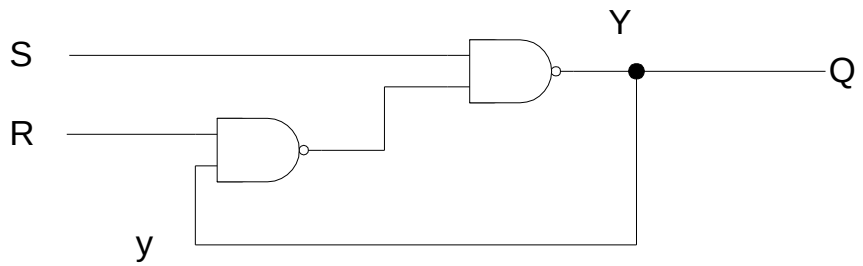
SR Latches

- Note: We can see the undesirable case when SR=11 and inputs change.
- Depending on the various delays and assuming SR=11 $\rightarrow$ SR=00…
  - If SR=11 -> SR=10 -> SR=00, we get stable state with output of 1.
  - If SR=11 -> SR=01 -> SR=00, we get stable state with output of 0.
- So the stable state is unpredictable.

curr state	next state				output
	SR=00	01	11	10	
y	Y	Y	Y	Y	
0	0	0	0	1	0
1	1	0	0	1	1

S'R' Latches

- We can analyze an S'R' latch using the previous technique:



S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(after $S=1, R=0$)
0	1	1	0	
1	1	1	0	(after $S=0, R=1$)
0	0	1	1	

- Equations derived for secondary variable (same equation for output):

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

- Since we want to avoid the SR=00 situation, we can write:

$$Y = \overline{S} + Ry \quad \text{if} \quad S'R' = 0$$

S'R' Latches

- Can derive the transition table and the flow table:

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

$$Y = \overline{S} + Ry \quad \text{if} \quad S'R' = 0$$

curr state	next state SR=00	01	11	10	output
y	Y	Y	Y	Y	
0	1	1	0	0	0
1	1	1	1	0	1

curr state	next state SR=00	01	11	10	output
y	Y	Y	Y	Y	
a	b	b	a	a	0
b	b	b	b	a	1

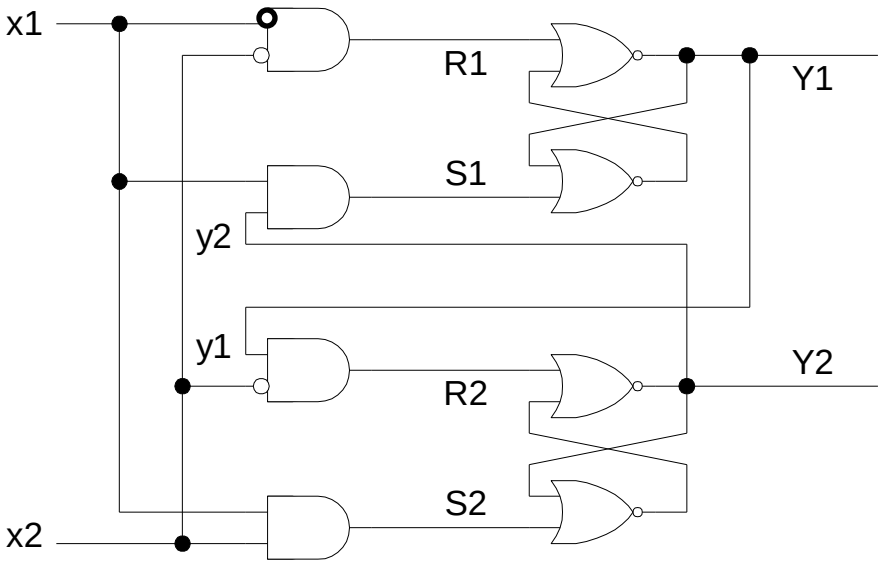
S'R' Latches

- Note: We can see the undesirable case when SR=00 and inputs change.
- Depending on the various delays and assuming SR=00 $\rightarrow$ SR=11…
  - If SR=00 -> SR=10 -> SR=11, we get stable state with output of 0.
  - If SR=00 -> SR=01 -> SR=11, we get stable state with output of 1.
- So the stable state is unpredictable.

curr state	next state				output
	SR=00	01	11	10	
y	Y	Y	Y	Y	
0	1	1	0	0	0
1	1	1	1	0	1

Analysis With Latches

- We might have asynchronous circuits with latches in them:



- We identify two inputs (x1,x2), two excitation variables (Y1,Y2), two secondary variables (y1,y2) and two latches.

Analysis With Latches

- Since we see latches, we obtain logic equations for the latch inputs:

$$\begin{aligned} S_1 &= x_1 y_2 \\ R_1 &= \overline{x}_1 \overline{x}_2 \end{aligned}$$

$$\begin{aligned} S_2 &= x_1 x_2 \\ R_2 &= \overline{x}_2 y_1 \end{aligned}$$

- Since we are working with latches, we should confirm that the latches do not ever enter the undesirable state (SR=11 for NOR, SR=00 for NAND).

- In our circuit, we have NOR latches, so we find:

$$S_1 R_1 = x_1 y_2 \overline{x}_1 \overline{x}_2 = 0$$

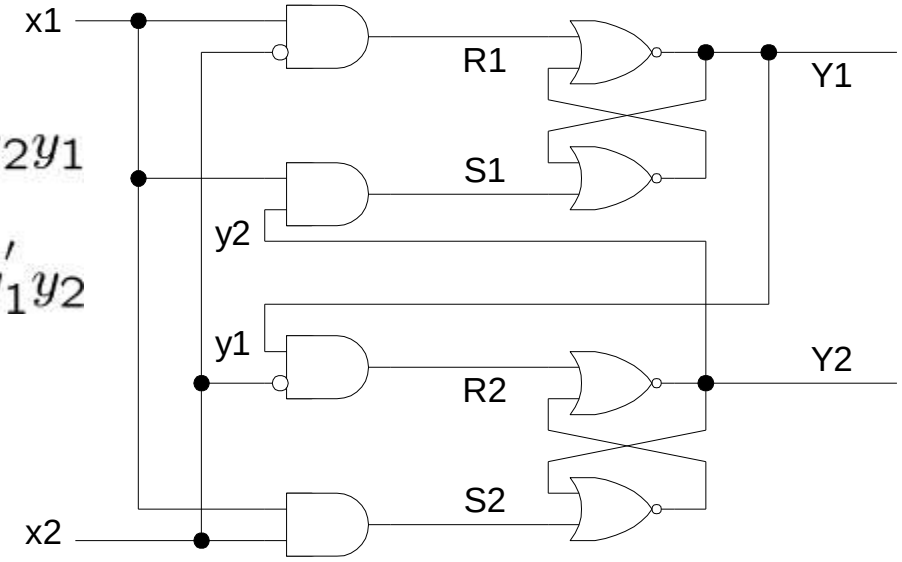
$$S_2 R_2 = x_1 x_2 \overline{x}_2 y_1 = 0$$

Analysis With Latches

- Derive the transition table.
- We need to find the excitation equations in terms of secondary variables and the circuit inputs.
- To do this, we need to use the latch equations:

$$Y_1 = S_1 + R_1' y_1 = x_1 y_2 + x_1 y_1 + x_2 y_1$$

$$Y_2 = S_2 + R_2' y_2 = x_1 x_2 + x_2 y_2 + y_1' y_2$$



Analysis With Latches

- Finally, use equations to derive transition table (could also find the flow table):

$$Y_1 = S_1 + R_1' y_1 = x_1 y_2 + x_1 y_1 + x_2 y_1$$

$$Y_2 = S_2 + R_2' y_2 = x_1 x_2 + x_2 y_2 + y_1' y_2$$

	x1x2			
curr state: y1y2	00	01	11	10
00	00	00	01	00
01	01	01	11	11
11	00	11	11	10
10	00	10	11	10

Y1Y2

Analysis Summary With Latches

- Label each latch output with $\mathbf{Y_j}$ and its feedback path with $\mathbf{y_j}$.
- Derive logic equations for latch inputs $\mathbf{S_j}$ **and** $\mathbf{R_j}$.
- Check of SR=0 for NOR Latches and S'R'=0 for NAND Latches. If not satisfied, the circuit may not work correctly.
- Create logic equations for latch outputs $\mathbf{Y_j}$ using the known behavior of a latch (**Y=S+R'y** for NOR Latches and **Y=S'+Ry** for NAND Latches).
- Construct a transition table using the logic equations for the latch outputs and circuit stable states.
- Obtain a flow table, if desired.

Textbook

- Analysis of asynchronous circuits in covered in Chapter 9, Section 9.2 and 9.3 of the course textbook (9.3 covers analysis with latches at outputs).

Asynchronous Circuit Design

- Similar procedure to synchronous circuit design, but with some added complexities due to the asynchronous part…
- Given verbal problem description:
  - Obtain a primitive flow table (one stable state per row) from problem description.
  - Reduce the flow table to get a smaller flow table with less states.
  - Perform state assignment (need to avoid race conditions) to obtain a transition table.
  - Obtain next state and output equations (need to avoid hazards and glitches).
  - Draw circuit (with or without latches).

Design Example

- Consider a circuit with two inputs, D and G and one output, Q. Output Q follows D with G=1, otherwise Q holds its value.
  - Assume fundamental mode operation – only one input changes at a time.

state	Inputs D	G	Output Q	Behavior : trnsfer D to o/p if G is 1; retain D value if G →0	
a	0	1	0	Transfer D to Q	
b	1	1	1	Transfer D to Q	
c	0	0	0	Keep previous Q=0; after a or d	
d	1	0	0	Keep previous Q=0; after c	
e	1	0	1	Keep previous Q=1; after b or f	
f	0	0	1	Keep previous Q=1; after e	

state	Inputs D	G	Output Q	Behavior : trnsfr D to o/p if G is 1; retain D value if G →0
a	0	1	0	Transfr D to Q
b	1	1	1	Transfr D to Q
c	0	0	0	Keep previous Q=0; after a or d
d	1	0	0	Keep previous Q=0; after c
e	1	0	1	Keep previous Q=1; after b or f
f	0	0	1	Keep previous Q=1; after e

- Note: Outputs depend only on state (Moore-like):

curr state	next state				output Q
	DG=00	DG=01	DG=10	DG=11	
a	c	(a)	-	b	0
b	-	a	e	(b)	1
c	(c)	a	d	-	0
d	c	-	(d)	b	0
e	f	-	(e)	b	1
f	(f)	a	e	-	1

Design Example – Primitive Flow Table
ONLY ONE STABLE STATE PER ROW

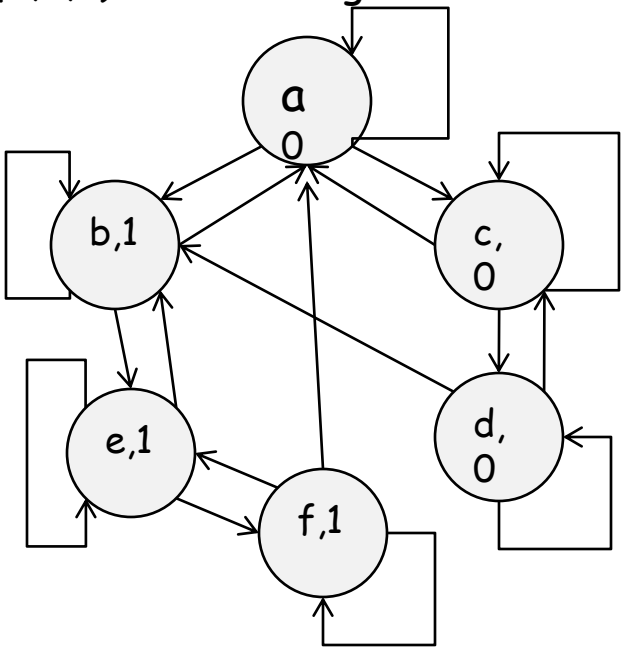
- Note: Some unspecified entries due to the fundamental mode assumption (e.g., in state a, DG=01, so we never go from DG=01 -> DG=10)…

Design Example – Reduced Flow Table

- For the moment, assume that the following flow table will also work for the verbal problem description – assume (a,c,d) and (b,e,f) can be merged.

- Original flow table:

curr state	next state				output Q
	DG=00	DG=01	DG=10	DG=11	
a	c	(a)	-	b	0
b	-	a	e	(b)	1
c	(c)	a	d	-	0
d	c	-	(d)	b	0
e	f	-	(e)	b	1
f	(f)	a	e	-	1



- Reduced flow table:

curr state	next state				output Q
	DG=00	DG=01	DG=10	DG=11	
a	(a)	(a)	(a)	b	0
b	(b)	a	(b)	(b)	1

Design Example - State Assignment and Transition Table

- We only have two states, so we can let a=0, and b=1.
- Our transition table becomes:

curr state (y)	next state (Y)				output Q
	DG=00	DG=01	DG=10	DG=11	
0	0	0	0	1	0
1	1	0	1	1	1

Design Example - Logic Equations

- We can make K-Maps to determine excitation variables (Y) and output (Z) in terms of circuit inputs and secondary variables (y):

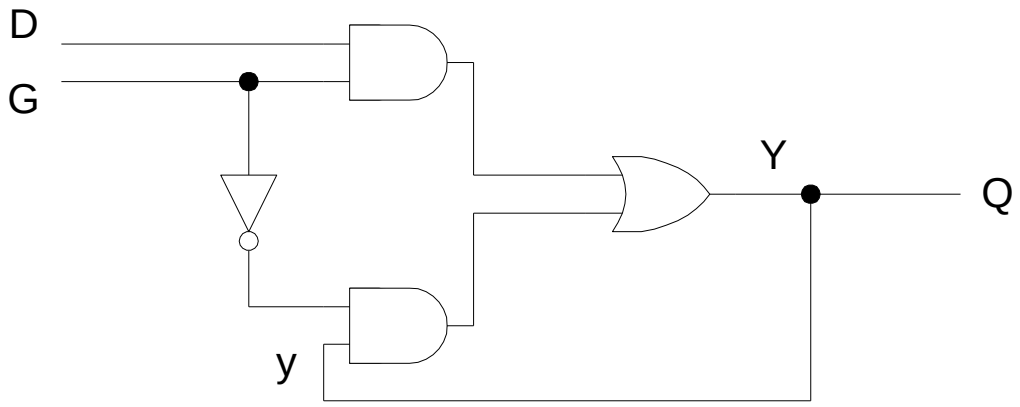
y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$$Y = DG + G'y$$

- Output equal to the secondary (state) variable.

Design Example - Circuit

- Can finally draw the circuit:

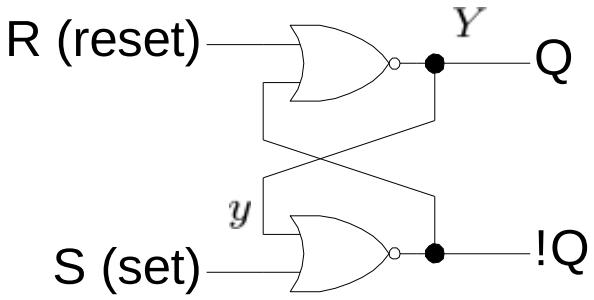


Implementation Using Latches

- We can also implement asynchronous circuits using latches at the outputs.
- Given the map for each excitation variable Y, derive necessary equations for S and R of a latch to produce Y.
- Derive Boolean equations for S and R.
  - Need to make sure the S and R never have equal (potential problem in Latch).

Implementation Using Latches – SR Latch Excitation Table

- Recall how a SR Latch (NOR) works:



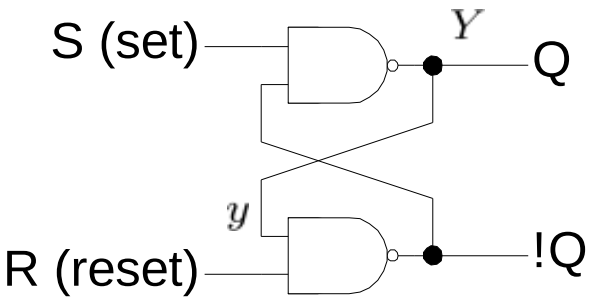
S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	(after $S=1, R=0$)
0	1	0	1	
0	0	0	1	(after $S=0, R=1$)
1	1	0	0	

- Assuming we never have the SR=11 case. Can write **excitation table**:

S	R	y	Y
0	X	0	0
1	0	0	1
0	1	1	0
X	0	1	1

Implementation Using Latches – S'R' Latch Excitation Table

- Recall an S'R' Latch (NAND) works:



S	R	Q	$\overline{Q}$	
1	0	0	1	
1	1	0	1	(after $S=1, R=0$)
0	1	1	0	
1	1	1	0	(after $S=0, R=1$)
0	0	1	1	

- Assuming we never have the SR=00 case. Can write **excitation table**:

S	R	y	Y
1	X	0	0
0	1	0	1
1	0	1	0
X	1	1	1

Implementation Using Latches

- Consider our example again, and assume we want to use a S'R' Latch:

y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

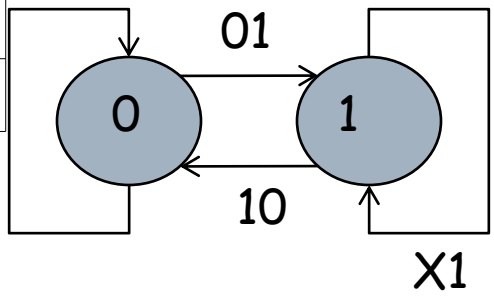
$$Y = DG + G'y$$

- Need to figure out how to select S and R for the NAND Latch (while making sure never 0 at same time):

S	R	y	Y
1	X	0	0
0	1	0	1
1	0	1	0
X	1	1	1

y \ DG	00	01	11	10
0	1	1	0	1
1	X	1	X	X

$$S = (DG)'$$

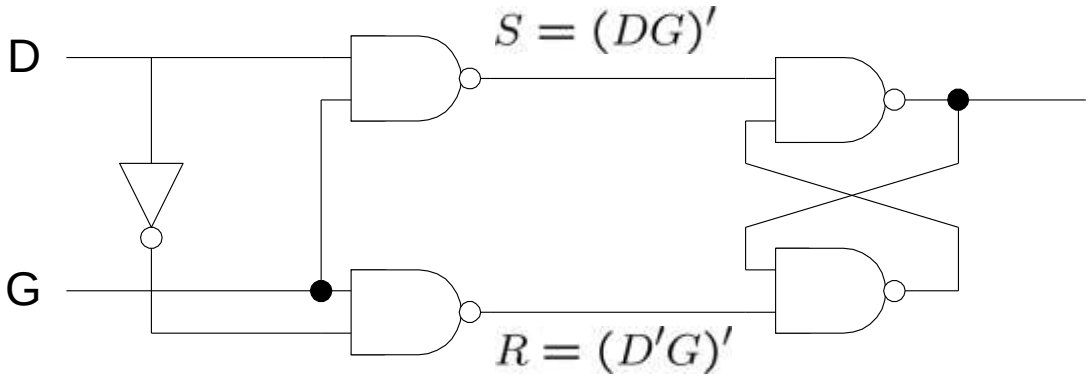


y \ DG	00	01	11	10
0	X	X	1	X
1	1	0	1	1

$$R = (D'G)'$$

Implementation Using Latches

- Can draw the circuit:



Textbook

- Design with and without latches is covered in Chapter 9, Sections 9.3 and 9.4 of the course textbook.

Output Assignment

- Flow and transition tables might have unspecified entries for circuit outputs.
  - This might be a result of the fundamental mode assumption.
  - This might be a result of unstable states.
- Note: output values always assigned for stable states!
- We should think about the correctness of these unspecified don't care output values…
  - We might temporarily pass through these values while transitioning from one stable state to another stable state.

Example

- Consider the following flow table with don't cares at some outputs (circuit has one input and one output):

curr state	next state		output	
	x=0	x=1	x=0	x=1
a	(a)	b	0	-
b	c	(b)	-	0
c	(c)	d	1	-
d	a	(d)	-	1

- We _might_ consider using the un-specified output values as don't cares in order to minimize the logic function for the output…

However…

- We need to be careful with output don't cares in asynchronous design.
- Consider start and stop STABLE STATES due to a change in input value.
  - If both stable states produce a 0 output, make output 0 instead of a don't care.
  - If both stable states produce a 1 output, make output 1 instead of a don't care.
  - If stable states produce different outputs, the output can remain a don't care and be used to find a smaller output circuit.
- We do this to avoid GLITCHES in the output (e.g., if the output should go 0->0 (or 1->1), it should remain 0 (or 1) during the transition through an unstable state.

Example

- Recall the flow table… If we consider possible transitions, we see that some of the output don't cares should be changed to 0 or 1 to avoid GLITCHES.

curr state	next state		output	
	x=0	x=1	x=0	x=1
a	(a)	b	0	-
b	c	(b)	-	0
c	(c)	d	1	-
d	a	(d)	-	1

curr state	next state		output	
	x=0	x=1	x=0	x=1
a	(a)	b	0	0
b	c	(b)	-	0
c	(c)	d	1	1
d	a	(d)	-	1

- The above changes will avoid temporary glitches at the outputs during transitions where the output should not change.

Textbook

- Output don't cares and avoiding glitches is covered in Chapter 9, Section 9.4 of the course textbook (near the end of the section).