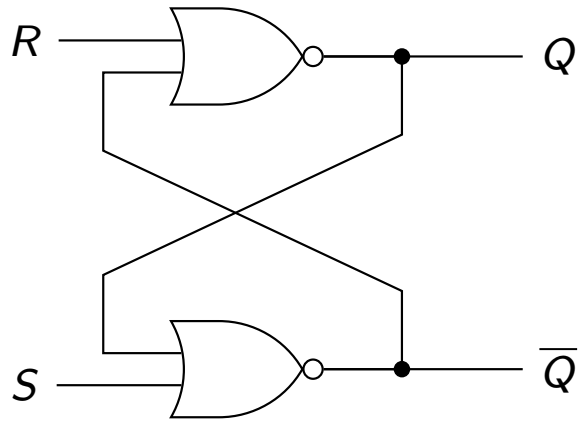
Asynchronous circuits

- We have seen combinational circuits. These are circuits which are logic functions which are written in terms of input variables and described by logic functions or truth tables.
- We have see clocked sequential circuits. These are circuits which involve combinational logic and flip flops. The behaviour of the circuit is controlled by a clock.
- There is another type of circuit that we might encounter, namely the *asynchronous circuit*.
- An asynchronous circuit can be defined as a circuit which exhibits the concept of *state* or *memory* much like a clocked sequential circuit. However, an asynchronous circuit *has no flip flops and no clock*.
- The concept of memory in an asynchronous circuit is a result of the combinational feedback paths (the loops) in the circuit and the delays involved in the circuit.
- In fact, asynchronous circuits were used prior to clocked sequential circuits. However, in many cases were replaced since their design and analysis is more difficult.

Asynchronous circuits

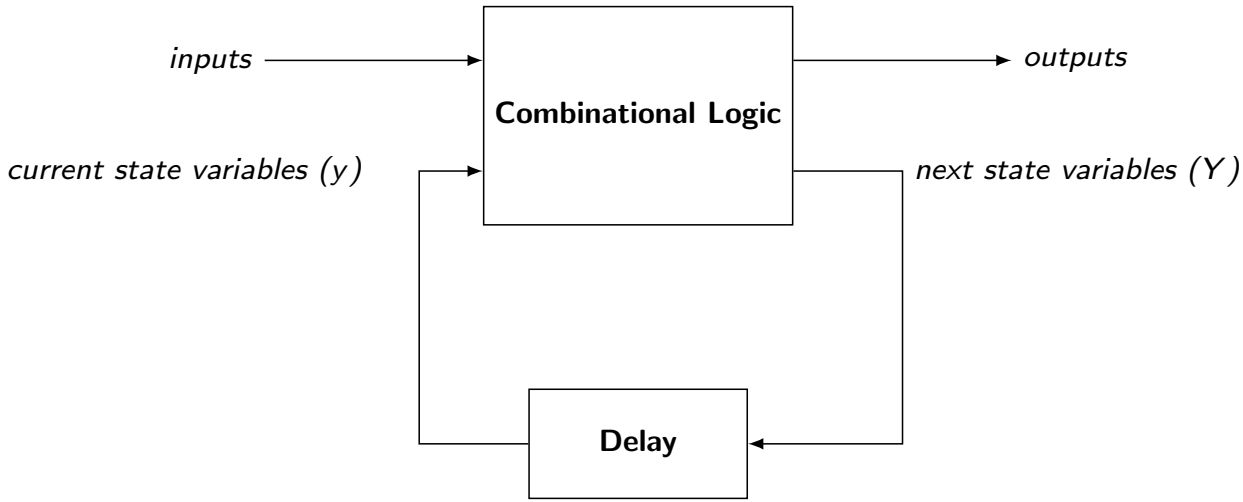
- We've seen an example already:



- The circuit is an SR latch which we already know has the concept of memory or state, but no clock. Our previous analysis was not really very formal.
- We need to figure out how to deal with such circuits including their analysis and their design.
- Asynchronous circuits are identified by the presence of latches and/or the presence of combinational feedback paths in a circuit diagram.

Asynchronous circuits

- A conceptual block diagram for a asynchronous circuit to try and identify things we can talk about:



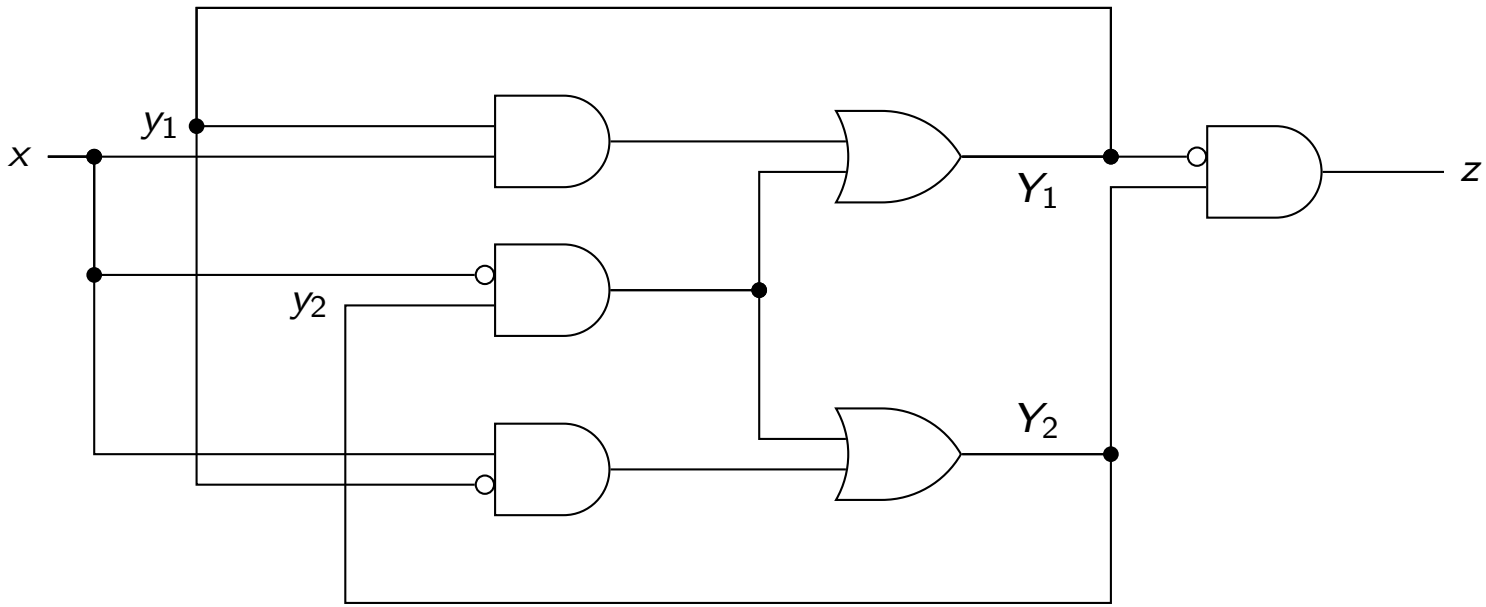
- The next state variables Y are often called the excitation variables. The current state variables y are often called the secondary variables.
- Basically, the excitation and secondary variables are the values at opposite ends of the same wire (which loops back in the circuit). However, they are separated by a delay and we can think of them as separate variables (equal to each other after a certain amount of delay) if we conceptually break the feedback loops.
- With excitation and secondary variables we can describe asynchronous circuits using tables similar to state tables (with minor modifications).

Asynchronous circuit analysis

- The analysis is similar to clocked sequential circuit analysis — we will derive tables that describes the behaviour of the circuit.
- The table that we derive, however, are not called state tables although they show similar information (that is, transitions between states as inputs change and circuit outputs). Rather, we will derive **transition tables** and **flow tables**.
- The difference is that a transition table shows the binary state assignments while the flow table is purely symbolic.
- For asynchronous circuits, when we derive these tables, we should indicate **stable states**.
- We require the definition of stability. For a given set of circuit input values, the circuit is **stable** if the circuit reaches **steady state** in which the excitation variables and secondary variables are equal to each other and unchanging, otherwise the circuit is unstable.

Asynchronous circuit analysis

- Consider the following circuit which we wish to analyze...



- In this circuit, there are no obvious latches present. However, there are certainly feedback paths in the circuit.
- We hypothetically break these paths in order to label the excitation and secondary variables.
- This circuit therefore has one input x, one output z, two excitation variables $Y_2\,Y_1$ and two secondary variables $y_2 y_1$.

Asynchronous circuit analysis

- We can write down equations for the excitation variables. These equations should be written in terms of the circuit inputs and the secondary variables (compared to clocked sequential design, this is equivalent to writing the next state functions in terms of the current state and circuit input variables):

$$\begin{aligned} Y_2 &= xy_1 + x'y_2 \\ Y_1 &= xy_1' + x'y_2 \end{aligned}$$

- We can also write an equation for the circuit output z. Again, we want to write this equation in terms of the secondary variables and circuit inputs (despite the fact that the circuit diagram makes it look like the output is a function of the excitation variables):

$$z = y_1'y_2$$

Asynchronous circuit analysis

- Using these equations we can determine the **transition table** for this circuit. Much like the state table, the **transition table** shows what the excitation variables (next state) will be given the secondary variables (current state) and the circuit inputs. It also shows the circuit outputs as a function of the secondary variables (our current state) and the circuit inputs.

Current state	**Next state** $(Y_2 Y_1)$		**Outputs** (z)	
$y_2 y_1$	$x = 0$	$x = 1$	$x = 0$	$x = 1$
00	**00**	10	0	0
01	00	**01**	0	0
11	**11**	01	0	0
10	11	**10**	1	1

- One significant difference between an asynchronous transition table and a state table is that in the transition table we **circle stable states**.
- Example... If we had $x = 0$ and $y_2 y_1 = 00$, then we would find that $Y_2 Y_1 = 00$ as well — the outputs of all gates in the circuit would be constant and not changing. However, if we changed $x = 0 \rightarrow 1$, we would find that $Y_2 Y_1$ are required to change from 00 to 10 which means that $y_2 y_1$ will also change to 10 (after some amount of delay). According to the transition table, we would end up in another stable situation in which $x = 1$ and $y_2 y_1 = 10$ (and $Y_2 Y_1 = 10$).

Asynchronous circuit analysis

- If we know the circuit always reaches a stable state when an input variable is changed, we can also observe that the output really only needs to be specified when the circuit is in a stable state.
- Therefore, we *could* consider rewriting the transition table as follows:

Current state	**Next state** $(Y_2 Y_1)$		**Outputs** (z)	
$y_2 y_1$	$x = 0$	$x = 1$	$x = 0$	$x = 1$
00	**00**	10	0	–
01	00	**01**	–	0
11	**11**	01	0	–
10	11	**10**	–	1

  This indicates that when the circuit is unstable, the output z is not relevant and can potentially change whenever it wants to just as long as it reaches a correct value upon entering another stable state.
- (There could, however, be reasons why the value of the output z is specified in certain unstable situations). We might touch on this later...

Asynchronous circuit analysis

- Because we had two feedback paths, we had the potential for 4 states represented by the binary values 00, 01, 10 and 11. We can "undo" this binary assignment to get a symbolic table (known as the flow tables):

Current state	**Next state** $(Y_2 Y_1)$		**Outputs** (z)	
$y_2 y_1$	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	$\boxed{a}$	d	0	–
b	a	$\boxed{b}$	–	0
c	$\boxed{c}$	b	0	–
d	c	$\boxed{d}$	–	1

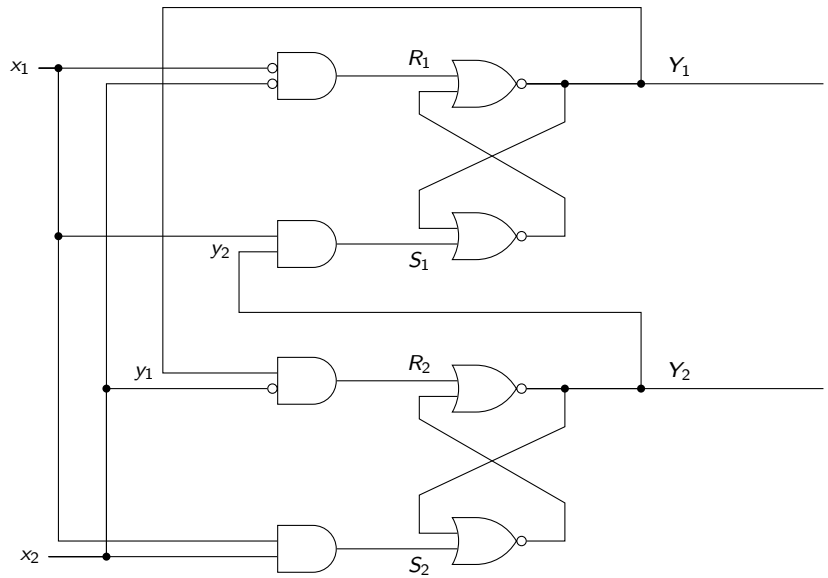
- Note that stable states are still boxed in the flow table.
- Just a comment... We could take the flow table and draw a state diagram if we wanted to...

Asynchronous circuit analysis

- Summary... Given a circuit with feedback paths, analysis consists of...
  - Identifying the feedback paths, hypothetically breaking these paths (turning the circuit — at least hypothetically — into a combinational circuit);
  - Labelling the excitation and secondary variables;
  - Deriving equations for the excitation variables and output variables in terms of the circuit inputs and secondary variables;
  - Deriving a transition table being careful to box the stable states;
  - Deriving a flow table by undoing the state assignment from the transition table.

Asynchronous circuit analysis — latches present

- It might be that we can easily identify latches in a circuit and, further, that it is the latch outputs that feedback around in the circuit to the latch inputs — in some ways, this makes the circuit look much more like a clocked sequential circuit.
- Example...



- This circuit has two input x_2x_1, two excitation variables Y_2Y_1 (latch outputs) and two secondary variables y_2y_1.
- Since we have identified two latches (in this case SR latches), we label the latch inputs.

Asynchronous circuit analysis — latches present

- Since we see latches, we can write down equations for the latch inputs in terms of the circuit inputs and secondary variables.
- For our circuit we get...

$$\begin{aligned} S_1 &= x_1 y_2 \\ R_1 &= x_1' x_2' \\ S_2 &= x_1 x_2 \\ R_2 &= x_2' y_1 \end{aligned}$$

- The excitation variables are the latch outputs. We can therefore use equations for the latch outputs written in terms of the latch inputs to ultimately write the latch outputs in terms of the secondary variables and circuit inputs:

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

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

Asynchronous circuit analysis — latches present

- We use the excitation equations to derive the transition table:

Current state y_2y_1	**Next state** (Y_2Y_1) $x_2x_1 = 00$	$x_2x_1 = 01$	$x_2x_1 = 11$	$x_2x_1 = 10$
00	**00**	**00**	10	**00**
01	00	**01**	11	**01**
11	00	**11**	**11**	01
10	**10**	**10**	11	11

- When working with latches, we might want to check to determine if the latches avoid the restricted state (for a SR latch is $SR = 11$). For this circuit, we see that $S_1R_1 = x_1y_2x_1'x_2' = 0$ and $S_2R_2 = x_1x_2x_2'y_1 = 0$. Therefore, these latches will not have both set and reset inputs at 1.

Asynchronous circuit design

- Similar the clocked sequential design, we can follow a sequence of steps to design a circuit from a verbal problem description.
- There are, however, some additional complexities that arise during asynchronous design. In addition, there are some other concepts we should exploit to make the design easier.
- When designing, we should make the **fundamental mode assumption**. An asynchronous circuit is said to be operating in fundamental mode if two conditions are true:
  - One **one circuit input** can change at any time;
  - The circuit is required to reach a stable state **prior** to changing a circuit input.
- When we design using the fundamental mode assumption, we should also derive what is called a **primitive flow table**. A primitive flow table is one in which there is **exactly one stable state per row**.

Asynchronous circuit design

- Example of a flow table which is not a primitive flow table...

Current state	**Next state** $(Y_2 Y_1)$			
	$x_2x_1 = 00$	$x_2x_1 = 01$	$x_2x_1 = 11$	$x_2x_1 = 10$
a	$\boxed{a}$	$\boxed{a}$	$\boxed{a}$	b
b	a	a	$\boxed{b}$	$\boxed{b}$

- Example of a primitive flow table...

Current state	**Next state** $(Y_2 Y_1)$	
	$x = 0$	$x = 1$
a	$\boxed{a}$	b
b	c	$\boxed{b}$
c	$\boxed{c}$	d
d	a	$\boxed{d}$

Asynchronous circuit design

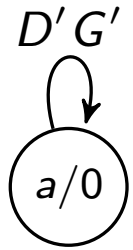
- Summary of the design procedure (pointing out some potential additional differences required for asynchronous circuits)...
- Given a verbal problem description...
  - Derive a state diagram (if desired) and a primitive flow table using the fundamental mode assumption;
  - Perform state reduction to obtain a smaller flow table with less states;
  - Perform state assignment to obtain a transition table. For asynchronous circuits, state assignment needs to be performed while avoiding critical races;
  - Derive equations for the excitation variables (next state) in terms of the secondary variables (current state) and circuit inputs. For asynchronous circuits, we should try to avoid combinational hazards;
  - Derive output equations in terms of the secondary variables and circuit inputs. For asynchronous circuits, we should try to avoid output glitches;
  - Draw the circuit.
- This procedure will result in an asynchronous circuit in which the presence of latches is not obvious. If so desired, we can amend the procedure to design explicitly using latches. In this case, rather that deriving equations for the excitation variables, we can derive equations for latch inputs in order to obtain the desired transitions from stable state to stable state.

Asynchronous circuit design

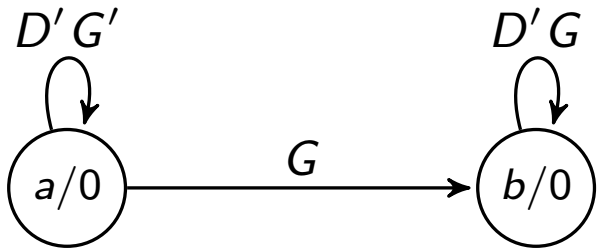
- Consider a circuit with two inputs D and G. The circuit has one output Q. When $G = 0$, the output Q maintains its current value, but when $G = 1$, the output Q is equal to the input D. Design an asynchronous circuit that exhibits this behaviour.
- Some comments...
  - Assume fundamental mode since this means from any stable state, we have less transitions to consider (only need to consider the transitions caused by changing each input variable);
  - To start, figure out one stable state and begin from there. Let an input change and transition to another stable state. Continue this procedure until you return to a previously created state (this is sort of like following a timing diagram with single inputs changing, the circuit reaching a stable state followed by another input changing).
  - Fill in any additional transitions required.
- For this problem, we can see that one stable state would be the situation in which $D = 0$, $G = 0$ and $Q = 0$.

Asynchronous circuit design

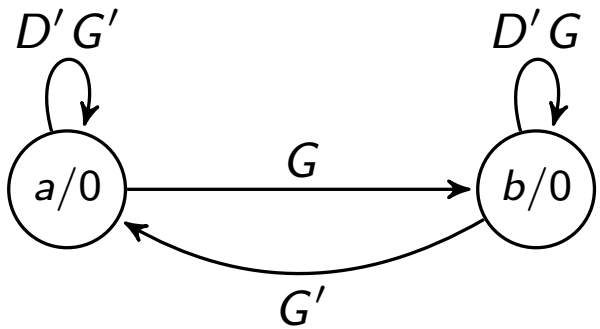
- Diagram with only the initial state... As long as $DG = 00$, the circuit is stable...



- Consider changing $G = 0 \to 1$. The problem specification says that Q should be equal to D (so $Q = 0$ since $D = 0$). According to our recommended procedure, since an input changed we should transition to another stable state. Observe that the new stable state also has a self loop which implies that it is a stable state so long as the inputs do not change...

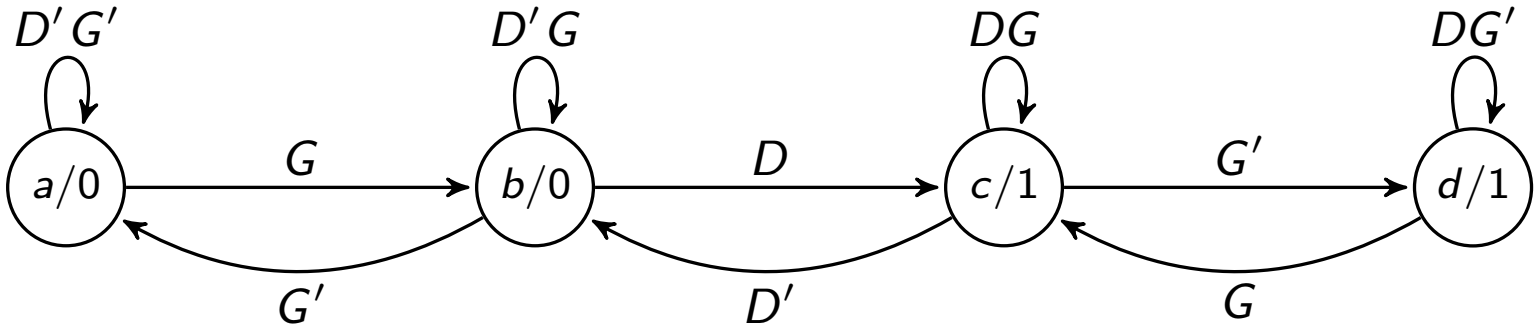


- Consider if we were in stable state b and the input $G = 1 \to 0$. We would go to stable state a so we can add that edge now.



Asynchronous circuit design

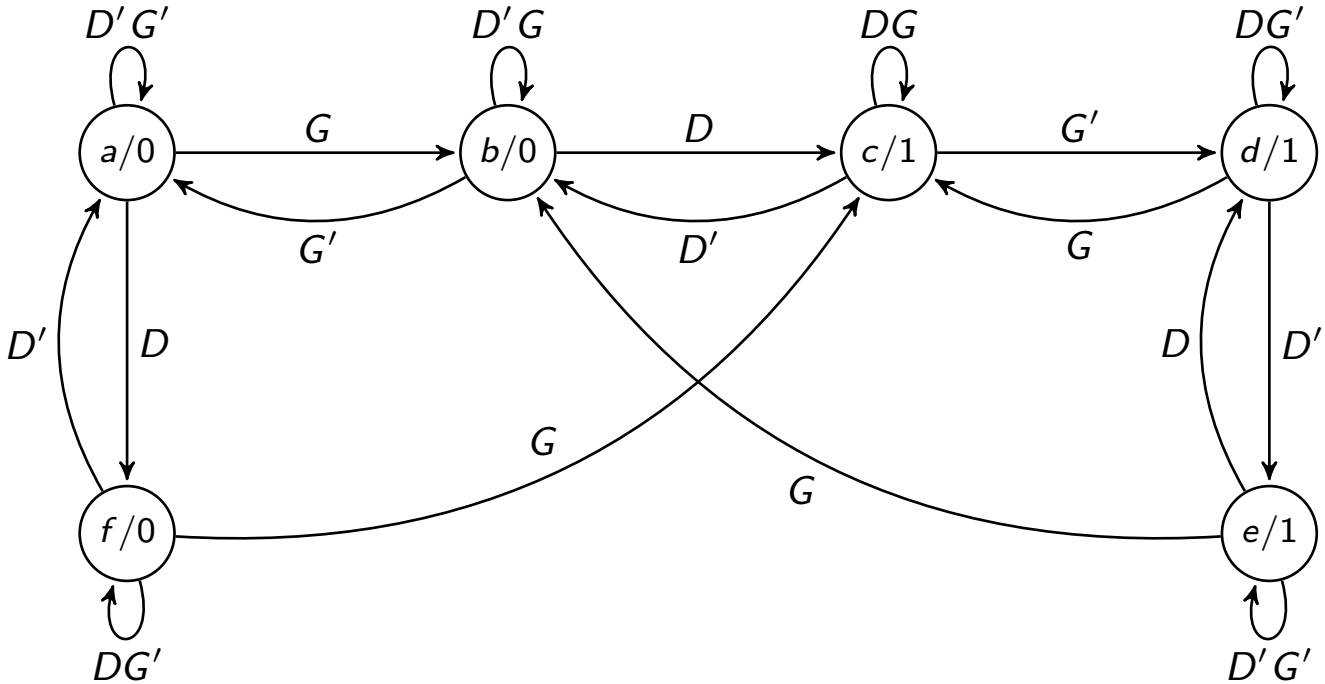
- Continue from stable state b changing one input at a time...



- We are not done, but we need to think carefully about how to proceed. For example, in stable state d we have $Q = 1$ and $DG = 10$. In other words, our circuit is "holding an output value of 1". If we change $D = 0 \rightarrow 1$ we will now have inputs $DG = 00$ and we have a stable state a in which $DG = 00$. However, stable state a produces output 0. But, from stable state d, if $D = 1 \rightarrow 0$, the output should still be 1. Therefore, we need at least one more stable state...

Asynchronous circuit design

- Our complete diagram...



- Interpretation: states a and f represent the scenario where $G = 0$ and $Q = 0$, but the input D is "flipping around". Similarly, states d and e represent a similar scenario, but $Q = 1$ despite D "flipping around". Finally, states b and c indicate the scenario where $G = 1$ and the output Q changes so that is always equals the value of D.

Asynchronous circuit design

- The (primitive) flow table...

Current state	**Next state**				**Output** (Q)			
	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$
a	$\boxed{a}$	b	–	f	0	–	–	–
b	a	$\boxed{b}$	c	–	–	0	–	–
c	–	b	$\boxed{c}$	d	–	–	1	–
d	e	–	c	$\boxed{d}$	–	–	–	1
e	$\boxed{e}$	b	–	d	1	–	–	–
f	a	–	c	$\boxed{f}$	–	–	–	0

- This flow table has a lot of don't care entries. This happen in the transition to the next state since we have assumed fundamental mode operation where inputs cannot change at the same time. This also happens in the value of the output since we only need to specify the value of the output when we are in a stable state.

Asynchronous circuit design

- The flow table should be reduced by merging equivalent states. The procedure for doing this is similar to state minimization with clocked sequential circuits (implication charts and merger diagrams). The major difference we encounter with asynchronous circuits is the large number of unspecified entries in the flow table. Succinctly stated, we can match unspecified entries with anything.
- The reduced flow table (we can merge a, b and f into one state and c, d and e into another state)...

Current state	**Next state**				**Output** (Q)			
	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$
a	$\boxed{a}$	$\boxed{a}$	c	$\boxed{a}$	0	0	–	0
c	$\boxed{c}$	a	$\boxed{c}$	$\boxed{c}$	1	–	1	1

- The next step is state assignment. Recall, when performing state assignment, we should avoid race conditions. There are no races here and we will skip that until we have a chance to talk about races.

Asynchronous circuit design

- Let state a be represented by 0 and let state c be represented by 1. This gives the transition table:

Current state (y)	**Next state** (Y)				**Output** (Q)			
	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$
0	[0]	[0]	1	[0]	0	0	–	0
1	[1]	0	[1]	[1]	1	–	1	1

- Since we only need 1-bit to represent the state, we have one excitation variable Y and one secondary variable y.
- Excitation equations:

$$Y = DG + G'y$$

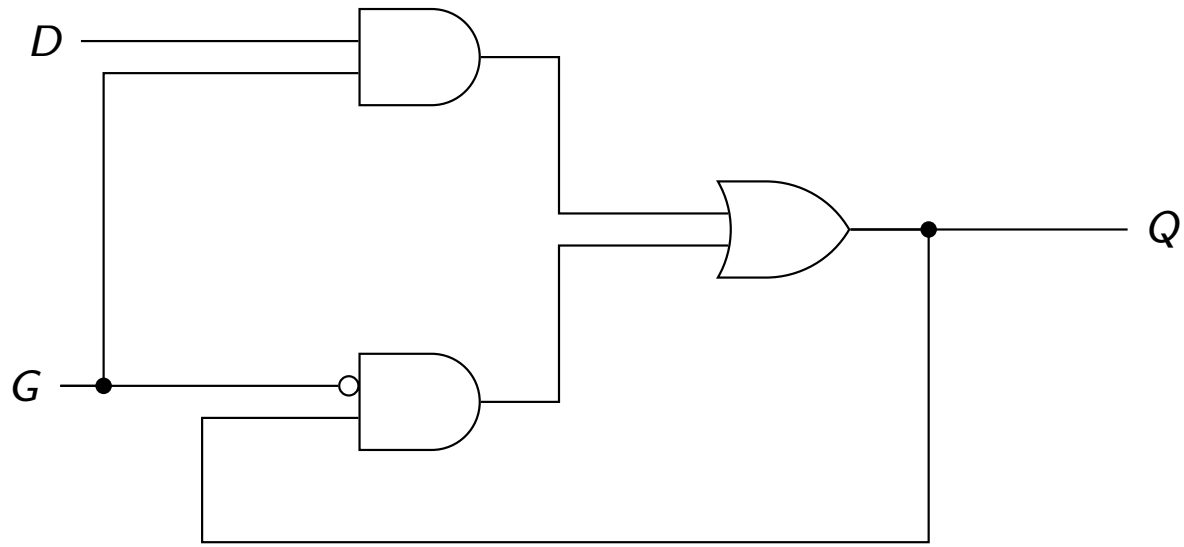
- Output equations:

$$Q = y$$

- In deriving the excitiation equations, we should consider whether or not our circuit will exhibit hazards. Further, we should consider whether or not our output equations will exhibit glitches. We won't worry about these considerations right now and return to that later.

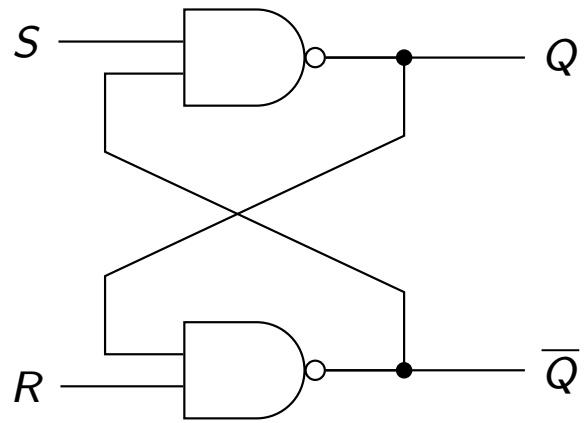
Asynchronous circuit design

- The circuit...



Asynchronous circuit design — using latches

- We could also consider using latches to hold the excitation variable values. Rather than deriving an equation for Y, we could derive equation for the latch inputs to obtain the desired state transitions.
- Assume we want to use $S'R'$ latches (**NAND** latches) (similar if using SR latches designed with **NOR** gates).
- Recall how $S'R'$ latch works... Notice the table is drawn a bit differently to help express how the latch output *changes*.



- $S'R'$ latch behaviour emphasizing how the output changes...

S	R	$Q(t) \rightarrow Q(t+1)$	
1	X	$0 \rightarrow 0$	*hold or reset*
1	0	$1 \rightarrow 0$	*reset*
0	1	$0 \rightarrow 1$	*set*
X	1	$1 \rightarrow 1$	*hold or set*

Asynchronous circuit design — using latches

- Our previous flow table...

Current state (y)	Next state (Y)				Output (Q)			
	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$	$DG = 00$	$DG = 01$	$DG = 11$	$DG = 10$
0	[0]	[0]	1	[0]	0	0	–	0
1	[1]	0	[1]	[1]	1	–	1	1

- Need to derive equations for the S and R latch inputs to get the desired transitions from $y \rightarrow Y$:

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

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

- Note that is a **NAND** latch. We should see if we properly avoid the situation where $SR = 00$. Test $S + R = (DG)' + (D'G)' = D' + G' + D + G' = 1$. Since $S + R$ is never equal to 0 we know that SR cannot be zero at the same time.

Asynchronous circuit design — using latches

- The circuit...