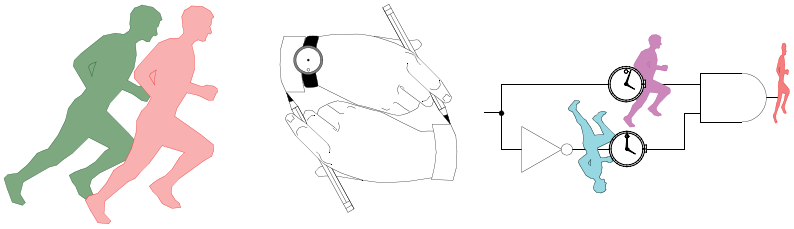
Asynchronous Circuits

Races, Cycles and Effect of Hazards

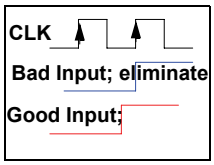


"*All circuits have problems, but asynchronous circuits have more problems*"

Feedback in Digital Circuits

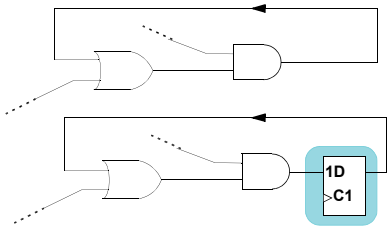
The Difference Between Asynchronous and Synchronous Feedback

- **Synchronous feedback** must wait for for the clock.
- Input signals must not change when the clock does.
- Always behave like the state table says



- Asynchronous feedback comes immediately with only gate delays.
- Inputs can come at any time.
- Circuits may not behave as the state table says.

FIG. 2-1 Illustrates synchronous and asynchronous feedback

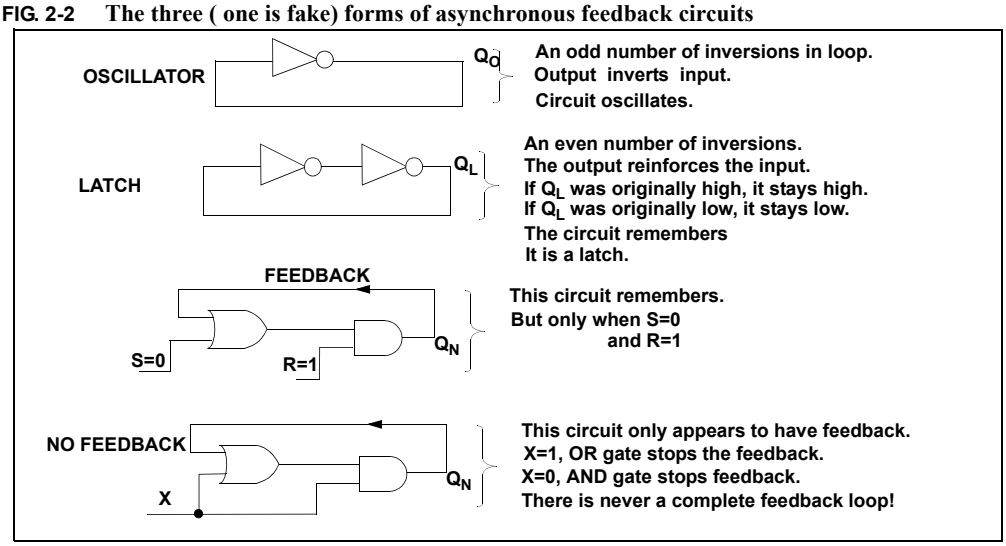


Asynchronous feedback.
the only delay in the feedback loop is the propagation delay of the gates.

Synchronous feedback.
The feedback is delayed until the active clock edge.

The 3 Forms of Asynchronous Feedback

FIG. 2-2 The three (one is fake) forms of asynchronous feedback circuits



Asynchronous feedback circuits *oscillate* or *remember*.

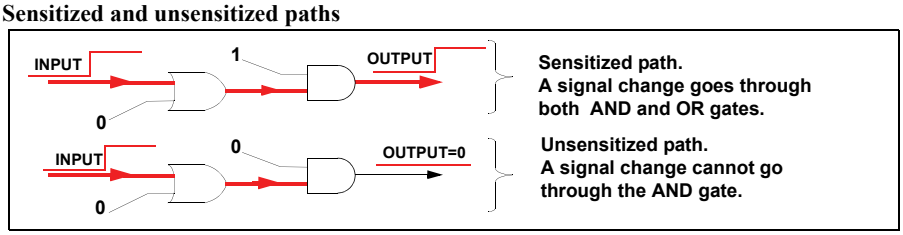
They *oscillate* if the feedback loop has an odd number of inversions.

They *remember* if the feedback loop has an even number of inversions.

Sensitized Paths

A signal path is *sensitized* if a change in its input changes its output.

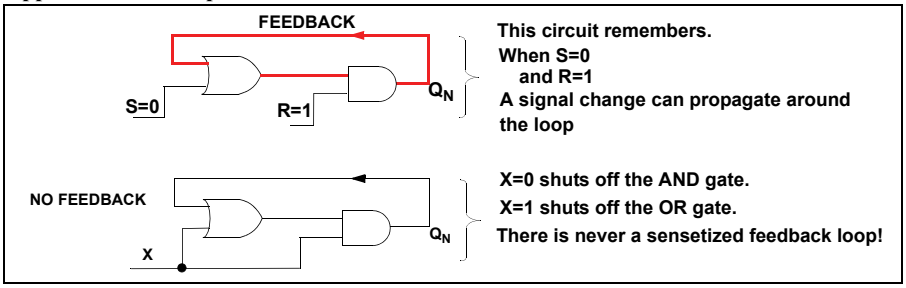
FIG. 2-3 Sensitized and unsensitized paths



A feedback path must be sensitized for the circuit to remember/oscillate

In practical circuits, the feedback is only *sensitized* part of the time.

FIG. 2-4 Apparent feedback path is never *sensitized*

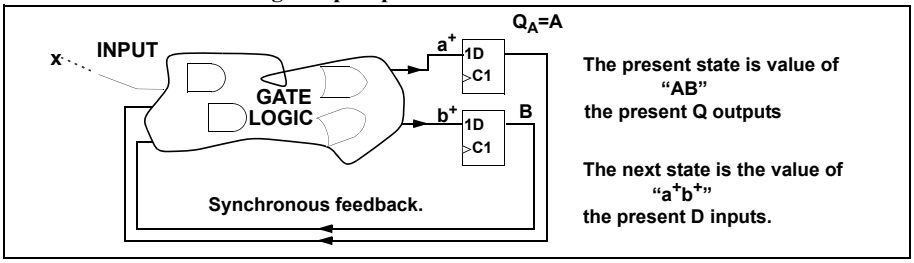


Analysis of Asynchronous Circuits

Review of Synchronous State Machines

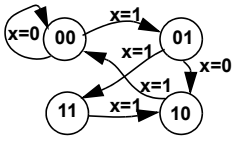
The state is the collective output of all flip-flops and latches.
The next state is the state after the clock edge.

FIG. 2-5 A state machine using D flip-flops.



A state machine is defined by its *next state table.*

FIG. 2-6 State Graph and Next State Table for a Mythological Circuit.



State AB	Next State $a^+b^+c^+$ x=0	x=1
00	00	01
01	10	11
11	10	10
10	11	00

The next state is different for different inputs.

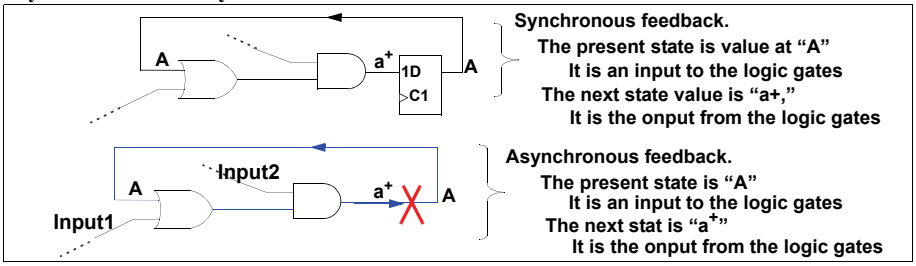
State Variables For Asynchronous Circuits

In synchronous circuits (with D flip-flops)
- Each flip-flop remembers one bit.
- Each flip-flop holds a state variable.
- The present state is the flip-flop output(s) "A"
- The next state is the the D input(s), "a^+"

In asynchronous circuits
- Each feedback loop latches one bit
- Each feedback loop holds a state variable.
- The state is the value fed back to the input.
- The next state is value out of the gate.

In Asynchronous circuits the state is value on the feedback lead.
- Mark an X on the wire where you want to read the state.
- Write "A", the present state on the output side.
- Write "a^+", the next state, on the input side.

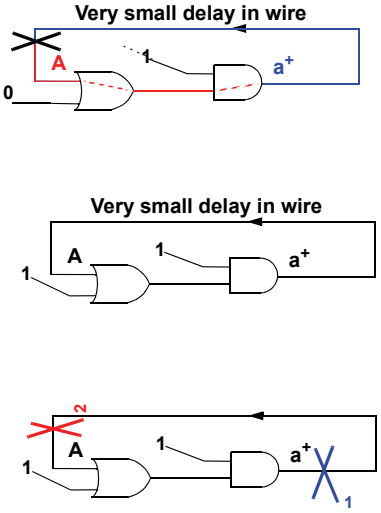
FIG. 2-7 Synchronous and asynchronous state variables



In steady state, the asynchronous present state = next state, $A = a^+$
On an input change, a^+ may change.
This will immediately change A and give a new state.

Relation Between "A" and "a+"

FIG. 2-8 Delay separates a^+ and A



Asynchronous feedback model.

a^+ comes out of circuit.
After a short delay it reenters the circuit as A

A travels through the sensitized path
and emerges as a^+.

The state A is continuously regenerated by a^+
a^+ is continuously recalculated from A and inputs

An input change may change a^+.
After the wire delay this will change A.
A will stay in this new state
until an input changes a^+ again.

We usually put an X on the schematic to show where a+ changes to A.
Should the X be at X_1 or X_2?*

* The X is thinking point where a^+ changes to A. It can be anywhere in the feedback path

Where To Put The Loop Breaks

FIG. 2-9 Break each *independent* feedback loop with an X (push button).

An independent feedback loop can store one bit.

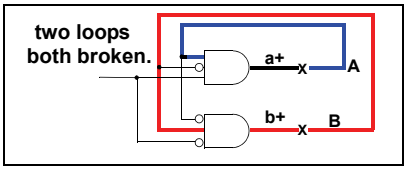


FIG. 2-10 Sometimes two independent loops are really one

Then cut them with one X.

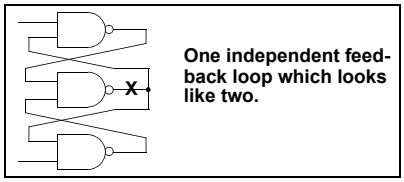
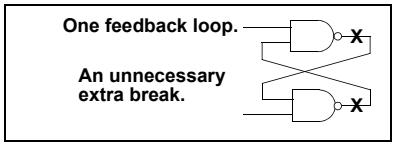


FIG. 2-11 Extra breaks:

Analysis will still work with several breaks in one loop.

Adds unnecessary states, make more work.

When in doubt, use extra breaks.



Analysis of an Asynchronous Circuit

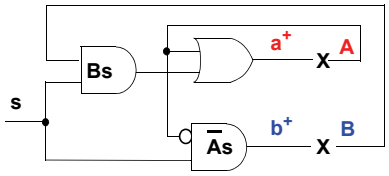
Tracing the Signals

A typical analysis:

1. Break the feedback loops at the appropriate places, A and B in FIG. 2-12.
2. Write equations relating the values on the input side of the break (a^+ and b^+) with values on the output side (A, B) and the inputs (s).

   That is $a^+ = f(A,B,s)$,
   $b^+ = g(A,B,s)$.
3. Make a state table.

FIG. 2-12 A Sample Circuit For Analysis.



$a^+ = f(A,B,s) = A + s\,B$

$b^+ = g(A,B,s) = \overline{A}s$

State A B	Next State a^+ b^+ s=0	s=1
00	00	01
01	00	11
11	10	10
10	10	10

1) Break the circuit at A and B. 2) Write equations for a^+,b^+ 3) Make a state table showing the next states for each state

FIG. 2-13 Tracing State Changes on the Sample Circuit.

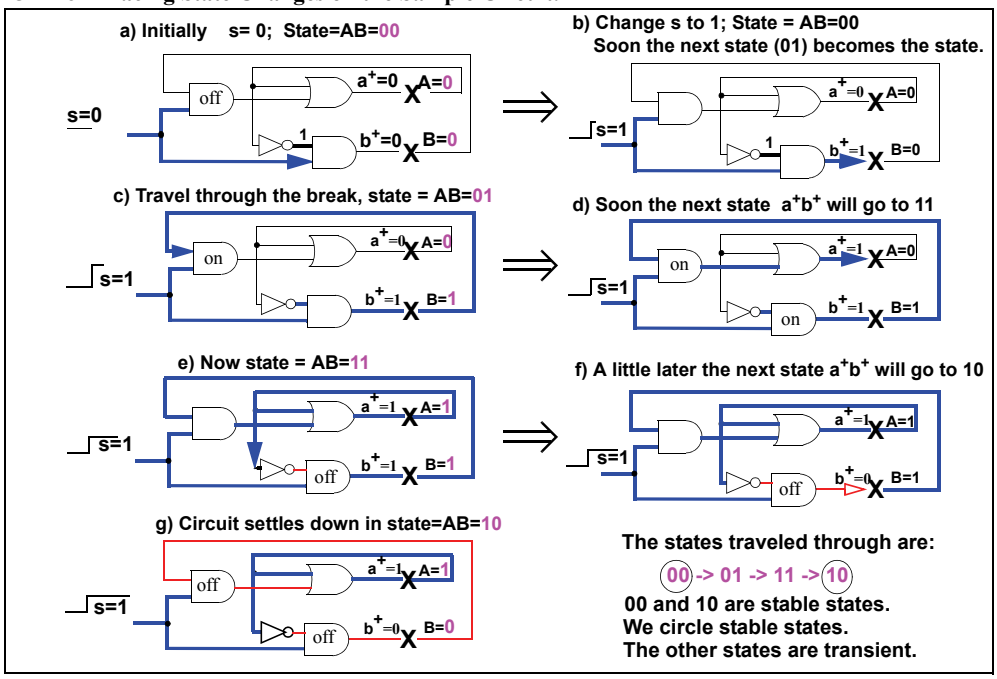


FIG. 2-14 Tracing State Changes on the Next State Table.

State A B	Next State a^+ b^+ s=0	s=1
00	(00)	01
01	00	11
11	10	10
10	(10)	(10)

Stable states:
where next state = present state.
are circled

Transient intermediate states:
all other states

Analysis

Start and end on stable states.

Travel on the next state side of table.

Path through next states is the same as the path through the present states.

States traveled through when s rises 0 1.

(00) -> 01 -> 11 -> (10)

Delays and the Sample Circuit

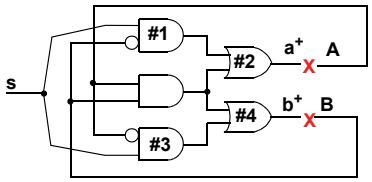
- The sequence of state changes is independent of the circuit delays.
- With random delays on all gates and all wires, the states would still change (00) -> 01 -> 11 -> (10)
- Be sure s does not change again until the final stable state is reached.
- This circuit is *delay Independent*.

Not all circuits are this well behaved.

Circuits Where Delays Matter

Circuits with Races

FIG. 2-15 Circuit where state changes depend on delays



Next-State Equations

$a^+ = f(A,B,s) = A\,B + s\cdot\overline{B}$

$b^+ = g(A,B,s) = A\,B + s\cdot\overline{A}$

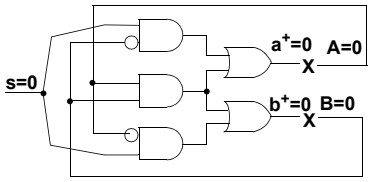
Substitute values for A, B and s into the next-state equations, to generate state table data.

State A B	Next State a^+ b^+ s=0	s=1
00	00	11
01	00	01
11	11	11
10	00	10

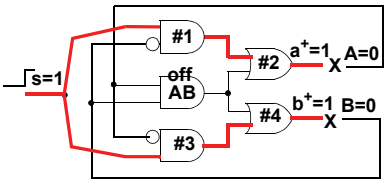
- This circuit does not always behave like the state table
- The sequence of states passed through depends on delays
- We will see:
  If #1 and #2 are faster, ends in state 10
  if #3 and #4 are faster, end in state 01,
  if they are equal, end in state 11.
- The circuit has a *race*.

FIG. 2-16 Analysis of a Racy Circuit

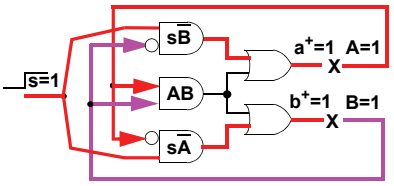
a) Initially s=0; State=AB=00



b) s changes s=1,
the next state $a^+b^+ = 11$
But only if the delay through gate #1 and #2
equals the delay through #3 and #4



c) AB = 11 feeds back to three gates.



d) The feedback blocks the output on $s\bar{B}$ and $s\bar{A}$
but reestablishes it through AB.
State change is 00 -> 11

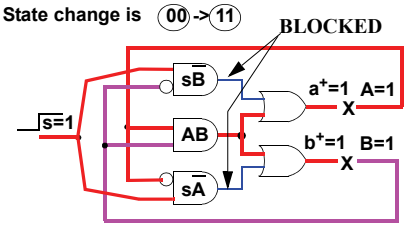
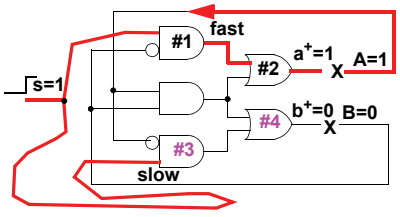


FIG. 2-17 The Analysis of Racy Circuit With Fast Upper Gates.

d) If gates #3 and #4 are slow,
after s changes s=1, next state $a^+b^+ = 10$.



e) AB = 10 will feed back to give a stable state
00 -> 10

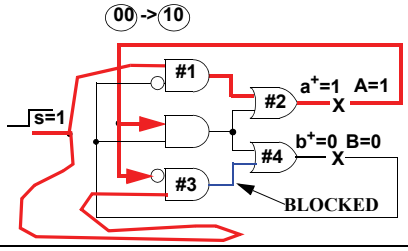
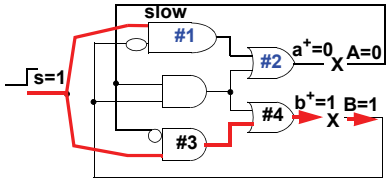


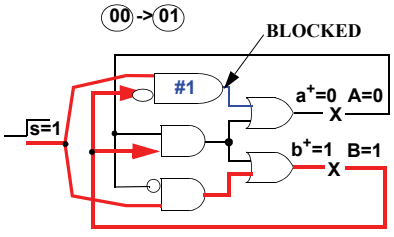
FIG. 2-18 Analysis With Slow Upper Gates

f) If gates #1 and #2 are slow,

after s changes s=1, next state $a^+b^+ = 01$.



g) AB = 01 will feed back to give a stable state
00 -> 01



Analysis of A Racy Circuit Using A State Table

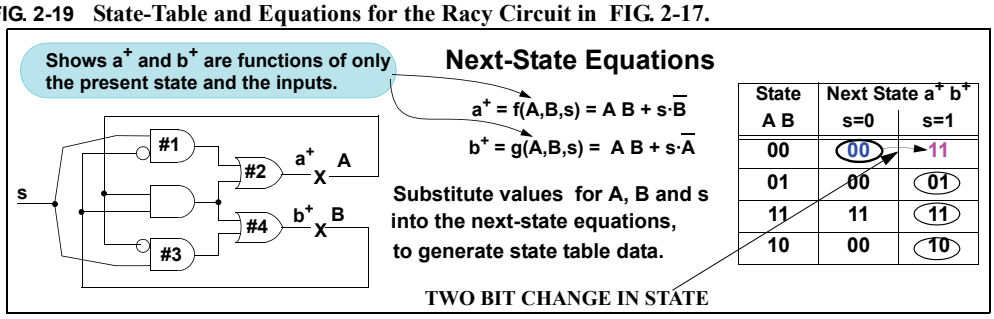
Get next-state equations from the circuit.
Use them to build a state table.

Note any two-bit change in the state.
when in state 00,
s goes 0 → 1
next state is 11

Identify races by:
A two-bit (or more) change in a state variable.

The possible results from the double-bit change is shown in FIG. 2-20.

FIG. 2-19 State-Table and Equations for the Racy Circuit in FIG. 2-17.



State A B	Next State a+ b+ s=0	s=1
00	00	11
01	00	01
11	11	11
10	00	10

FIG. 2-20 The Three Possible Outcomes of the Race in the Racy Circuit.

Path in the state-table if delays are equal.

State A B	Next State a+ b+ s=0	s=1
00	00	11
01	00	01
11	11	11
10	00	10

Path when delay for A (via gates #1 and #2) is slower.

State A B	Next State a+ b+ s=0	s=1
00	00	11 → 01
01	00	01
11	11	11
10	00	10

A is so slow it stays 0 until it is too late. (The stable state 01 is reached).

Path when delay for B (via gates #3 and #4) is slower.

State A B	Next State a+ b+ s=0	s=1
00	00	11 → 10
01	00	01
11	11	11
10	00	10

If gates #1 and #2 are fastest, the stable next state will be AB=10.

If gates #3 and #4 are fastest, the stable next state will be AB=01.

With equal delays, the stable next state will be AB=11.

Races Have a Simultaneous Two-Bit Change in the State Variables

Look for double-bit change in a state variables, and you will find the races.

Noncritical, and Other Non-important Races.

Not all races are a problem:

Noncritical Races

Some races take different paths but end in the same final stable state.
These are called *noncritical races*. FIG. 2-21 and FIG. 2-22

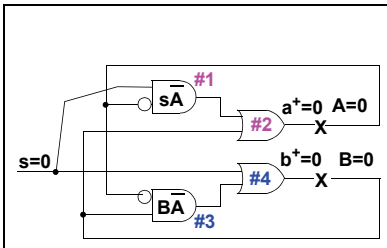
Unused Races ("Who Cares" Races)

Some races are in a part of the state table which is never used. FIG. 2-23

Races To Equivalent States (States that could be merged)

The final states of the race might be equivalent.
The machine would externally behave the same no matter how the race came out.

FIG. 2-21 A Noncritical Race With Three Paths.



Next-State Equations

$a^+ = f(A,B,s) = B + s\cdot\overline{A}$

$b^+ = g(A,B,s) = s + B\cdot\overline{A}$

State A B	Next State a^+ b^+ s=0	s=1
00	00	11
01	11	11
11	10	11
10	00	01

A slow B slow

FIG. 2-22 The Three Possible Paths Individually Shown.

Path for equal delays.

State A B	Next State a^+ b^+ s=0	s=1
00	00	11
01	11	11
11	10	11
10	00	01

Gates #1 and #2 are slower.
Then A is slower than B

State A B	Next State a^+ b^+ s=0	s=1
00	00	11
01	11	11
11	10	11
10	00	01

A is slow so next state becomes 01

Gates #3 and #4 are slower.
Then B is slower than A

State A B	Next State a^+ b^+ s=0	s=1
00	00	11
01	11	11
11	10	11
10	00	01

The race is noncritical.
All paths lead to state 11

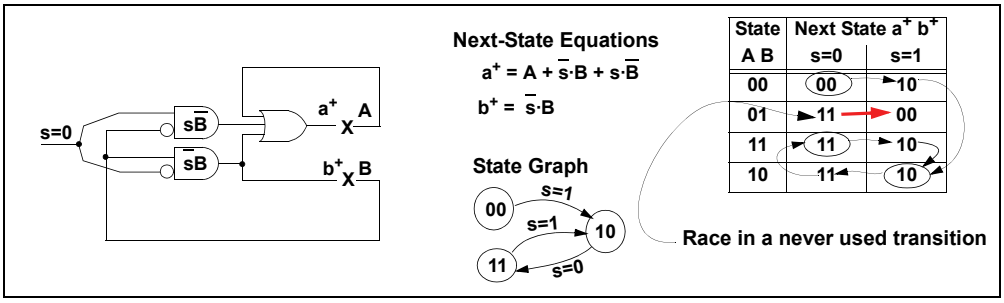
A "Who Cares" Race.

This Race is in state 01 as *s* changes ⟶.
The stable states and all the transitions between them are shown.

The "Who Cares" race is between unstable states.
Further, no useful transition goes through it.

The circuit does not use the transition.
No one cares if it has a race.

FIG. 2-23 A Circuit With a "Who Cares" Race.



Races Between Transient States Where One Cares!

A Race Between Transient States, But On a Path Between Stable States.

FIG. 2-23 showed a "who cares race" between unused transient states.

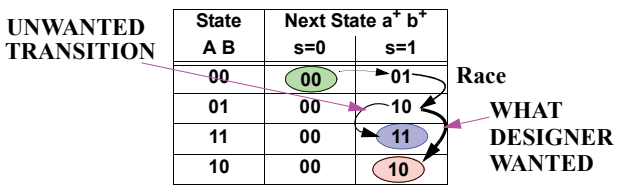
This race is between transient states, but should not be ignored.

It is an intermediate transition in the path between two stable states.-

00 → 01 → 10

The second transition from 01→ 10 has a double bit change.
The circuit may end up in the stable state 11 .

FIG. 2-24 Ra



Oscillations (Cycles) In State Tables.

Cycles **are oscillations**
They are transitions that once entered, never reach a stable state.

The Two Types of Cycles:

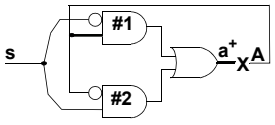
1. **The oscillation continue forever or until turned off.**
   These have:
   no stable state in the path of the cycle
   no races in the path that can reach a stable state.

2. **It oscillates a few cycles before falling into a stable state.**
   These have:
   a stable state in the column,
   a race in the path.[1]

Not covered in these short form notes.

1. For three or more state variables, also check the race can reach a stable state.

FIG. 2-25 A Cycle Which Will Continue Oscillating Forever.



Electronic Coin Tosser

No Stable States In s=1 Column

When s=1, It will oscillate forever.

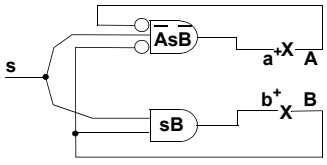
There are two feedback paths
The one through gate #2 oscillates (odd number of inverters).
The one through gate #1 is a latch. (even number of inverters).

The circuit will:
oscillate when s=1, and
latch the last output value when s -> 0.

State A	Next State a^+ s=0	s=1
0	0	1
1	1	0

FIG. 2-26 A Cycle, With Stable States In Column, But No Races, Will Continue Forever.

Stable States In the s=1 Column Without a Race



From the state table; there is a cycle.
From the circuit; the A loop oscillates.
The B loop stays fixed at B=0.
The A loop oscillating cannot make B=1
The A loop oscillating cannot reach a stable state.

State A B	Next State a^+ b^+ s=0	s=1
00	00	10
01	00	01
11	00	01
10	00	00

Asychronous State-Machines With Several Input Variables.

With only one external input. →

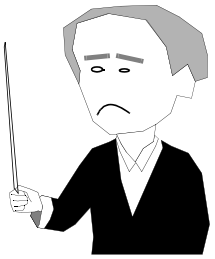
Single input asynchronous circuits have two types of races

1. Races between the state bits.
2. Races between a state variable and an input.
   These are called *essential hazards* and will not be discussed.

With two or more inputs. →

In addition, with multiple inputs, one can have-

3. Races between the two inputs



Multiple Asynchronous Inputs are Miserable

Two inputs races cannot be handled by this theory.

Theoreticians usually state-
Nearly simultaneous input changes are ***not allowed***.

Practical designers must force rationally behavior for double changes.
One might:
Design the machine to work properly for all states a race might enter.
Interlock the signals so they cannot change at the same time.
Feed the multiple input signals separately into a synchronous circuit.
See next course.

A Two-input State-Table With Races

Double input changes are *not allowed*.

The State-Table has 4 other races.

1. Start at (11) with sx=10 -> sx=11.
   This race leads into a cycle

2. Start at (01) with sx=00 -> sx=10.
   This wraps around the table.
   The final state must be stable.
   It may be any of
   (00) , (11) or (10).

FIG. 2-27 Two input async. circuit

State A B	Next State $a^+ b^+$ sx=00	sx=01	sx=11	sx=10
00	(00)	01 NO	10 NO	(00)
01	(01)	(01)	00 NO	10 NO
11	00	00 NO	00 NO	(11)
10	00	(10)	11 NO	10 NO

No means we don't allow two inputs to change at once.

FIG. 2-28

1)

AB	$a^+ b^+$ 00	01	11	10
00	00	01	10	00
01	01	01	00	10
11	00	00	00	
10	00	10	11	10

2)

AB	$a^+ b^+$ 00	01	11	10
00	(00)	01	10	(00)
01	(01)	(01)	00	10
11	00	00	00	(11)
10	00	(10)	11	10

A Two-input State-Table With Races (Cont)

Double input changes are *not allowed*.

The races continued:

3. Start at (10) with sx=10 -> sx=11.
   No race on first hop (10)
   2nd hop has a race, and
   leads into the same cycle as 1)

4. Start at (11) with sx=10 -> sx=00.
   This may end in either states
   (00) or (01)

FIG. 2-29 Two input async. circuit

State A B	Next State $a^+ b^+$ sx=00	sx=01	sx=11	sx=10
00	(00)	01	10	(00)
01	(01)	(01)	00	10
11	00	00	00	(11)
10	00	(10)	11	(10)

FIG. 2-30

3)

AB	$a^+ b^+$ 00	01	11	10
00	(00)	01	10	(00)
01	(01)	(01)	00	10
11	00	00	00	(11)
10	00	(10)	11	(10)

4)

AB	$a^+ b^+$ 00	01	11	10
00	(00)	01	10	(00)
01	(01)	(01)	00	10
11	00	00	00	(11)
10	00	(10)	11	(10)

5. State variable races with double input variable changes.
   State marked (01) NO NO [10] AB=01
   The double input change will make a mess of the circuit function.
   We take steps to make sure it will never happen (*not allowed*).
   Or (*next year*) follow all possibilities

Removal of Races

Race-free designs are done by careful state assignment.
Races are double-bit state-variable change.

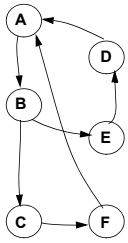
Thus states connected to another state
must differ in value by one-and-only-one bit

Adjacent Karnaugh map squares differ by one bit.
Assign states on a Karnaugh map.

Short summary

States joined by arrows should cuddle on a K-map.

FIG. 2-31 Assigning Bit-Patterns to States Using a Karnaugh Map.



Connected states must differ by only one bit

Adjacent Karnaugh map squares differ by one bit.

	0	1
00	000	001
01	010	011
11	110	111
10	100	101

	0	1
00	A	D
01	B	E
11	C	
10	F	E

Print the state names on map
place connected states
in adjacent squares.

Read the state assignment off the map

A = 000
B = 010
C = 110
F = 100
D = 001
E = 101

FIG. 2-32 Assigning Bit-Patterns to Difficult State Graphs By Adding States.

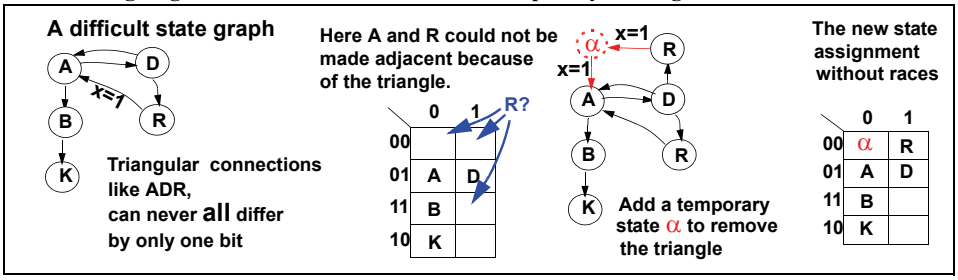
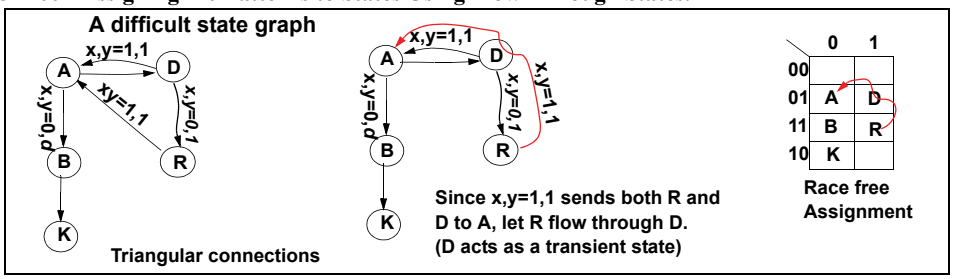


FIG. 2-33 Assigning Bit-Patterns to States Using Flow-Through States.



Adding extra state variables will eventually solve the race problem.
However it may make the circuit larger and slightly slower.

Find A Race Free Assignment for the State Table

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	R	A	R	B
A	A	A	D	B
B	A	--	--	B
C	A	C	R	--
D	D	C	R	D

(1) Circle stable states.

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	(R)	A	(R)	B
A	(A)	(A)	D	B
B	A	--	--	(B)
C	A	(C)	R	--
D	(D)	C	R	(D)

(2) Fill in the allowed (single input change) transitions from A to other stable states.

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	(R)	A	(R)	B
A	(A)	(A)	D	B
B	A	--	--	(B)
C	A	(C)	R	--
D	(D)	C	R	(D)

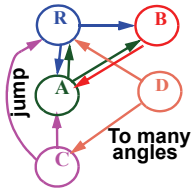
(3) Fill in the other transitions between stable states, for single column input changes.

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	(R)	A	(R)	B
A	(A)	(A)	D	B
B	A	--	--	(B)
C	A	(C)	R	--
D	(D)	C	R	(D)

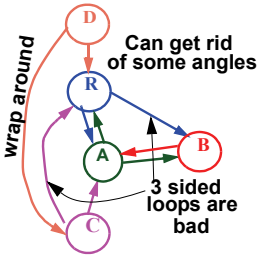
Finding A Race Free Assignment

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	R	A	R	B
A	A	A	D	B
B	A	--	--	B
C	A	C	R	--
D	D	C	R	D

(3) Sketch transitions on a state table. Try to make transitions horizontal or vertical.



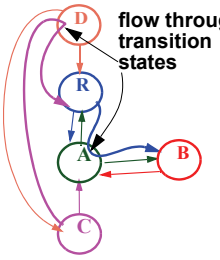
(4) Get rid of angles and jumps



Revise some next states to be flow-through states (two hops) to avoid races.

(5) Get rid of 3 sided loops.

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	R	A	R	~~A~~ B
A	A	A	~~D~~ R	B
B	A	--	--	B
C	A	C	~~D~~ R	--
D	D	C	R	D



(6) Move state locations to map

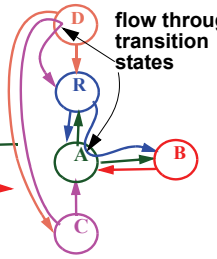
	0	1
00	D	
01	R	
11	A	B
10	C	

(7) Rotate map map to place R in 000

	0	1
00	R	
01	A	B
11	C	
10	D	

Race Free Assignments

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R	R	A	R	~~B~~ ~~A~~
A	A	A	~~D~~ R	B
B	A	--	--	B
C	A	C	~~R~~ D	--
D	D	C	R	D



	0	1
00	R	
01	A	B
11	C	
10	D	

(9) Fill in new transient states and race-free assignment.

State	Next State Input XY=00	Input XY=01	Input XY=11	Input XY=10
R=000	R=000	A=001	R=000	A=001
A=001	A=001	A=001	R=000	B=101
B=101	A=001	--	--	B=101
C=011	A=001	C=011	D=010	--
D=010	D=010	C=011	R=000	D=010

(8) Get race-free state assignment off map

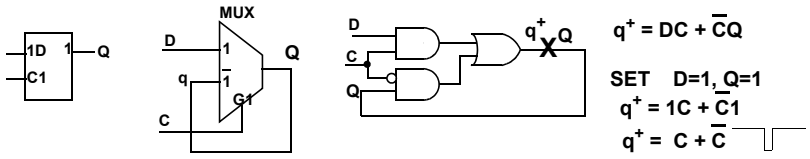
R=000
A=001
B=101
C=011
D=010

Hazards Can Poison Asynchronous Machines

Example: The Hazard in the Transparent D-Latch

The simple form of D-latch (transparent latch) is just a MUX.
The latch has a static-1 hazard when D,Q =1,1.
This glitch can feed back and latch itself.

FIG. 2-34

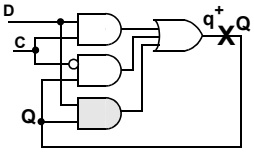


$q^+ = DC + \overline{C}Q$

SET D=1, Q=1
$q^+ = 1C + \overline{C}1$
$q^+ = C + \overline{C}$

As the clock falls
The 0 may get latched as Q=0
Even though D=1, and Q_{old} =1

To mask the hazard
Keep DQ =11 across
the change in C
$q^+ = DC + \overline{C}Q + DQ$



Hazard-free latch

$q^+ = DC + \overline{C}Q + DQ$

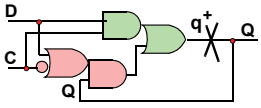
SET D=1, Q=1
$q^+ = 1C + \overline{C}1 + 1$
$q^+ = 1$

Common Hazard-Free D-Latch Circuit

Explanatory note

This D-latchcircuit is often used and is hazard free.
Analysis of the transistor circuits will show this is about 30% larger than the latch with the hazard, and smaller than the last circuit with the masking gate.

FIG. 2-35



$q^+ = DC + q(D+\overline{C})$

No hazard

Problems

1. Undesirable Properties Detectable From State Tables

Table 1: State Tables for despicable machines

Present State Q_1Q_2	Next State $Q_1^+Q_2^+$		Output Z	
	x = 0	x =1	x = 0	x = 1
00	00	01	0	1
01	01	00	0	0
11	00	11	0	0
10	11	10	1	0

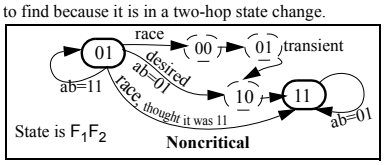
Present State F_1F_2	Next State $F_1^+F_2^+$			
	ab = 00	ab =01	ab = 11	ab = 10
00	11	01	00	00
01	11	10	01	01
11	11	11	11	01
10	10	11	00	10

These state tables have several problems.

The left table has a race, a cycle and a second race. The second race may be hard to find because it is in a two-hop state change.

The right table has three races.

a) Circle the stable states.
b) Use arrows on the state table to show the races
c) Draw a partial state graph to show each problem transitions. See example ->t
d) Indicate which graph you are using
e) State wheither each race is critical or noncritical.



2. Supernatural Problem

The following problem was posed in the sequential circuits lab in the University of Massachusetts at Amherst. It is allegedly based on a letter sent to Sherlock Holmes by Lamont Cranston.

January 18, 1892

My dear Mr. Holmes:

Some time ago I purchased an old house in which I am presently living. Unfortunately I found it to be haunted by two ghostly noises - a Ribald Singing and a Sardonic Laughter. As a result it is hardly habitable. There is hope however, for by actual testing I have found their behavior is subject to certain laws, obscure but infallible, and that they can be affected by my playing the organ or burning incense.

In each ten minute period, each noise is either sounding or silent - on or off. What each will do during the ensuing period depends only on what has happened during the immediately preceding period, in the following exact way.

The singing in the succeeding period will continue as before (sounding or silent) unless there was organ playing and no laughter. If there was organ playing and no laughter, then the singing will reverse (silent will become sounding, sounding will become silent).

As for the laughter, if there was incense burning, then the laughter will follow the singing of the previous period. Thus with incense burning, the laughter sounds if the singing sounded for the previous ten minutes. If however, there is no incense burning, the laughter will do the opposite of what the previous singing did.

At this minute of writing, the laughter and the singing are both sounding and I indeed fear for my sanity. Please tell me what manipulations of organ and incense I should make in order to make the house quiet and keep it so.

With kindest regards to Dr. Watson, I remain,

Your admiring friend,

Lamont Cranston

February 3, 1892

Dear Mr. Cranston:

Your problem is a most interesting one. Dr. Watson and myself spent an evening pondering your puzzle, and we believe we have a methodology which will offer you relief.

Following the work of the late Mr. George Boole in his book, "The Laws of Thought," we will use the symbol S for the state variable "singing," S will be 1 for the presence of singing or 0 for its absence. S+ will represent this state variable S during the forthcoming ten minute interval. In a similar way we will use L as a state variable for the laughter. Let the inputs to this ghastly system, the burning incense and the organ music, be represented by the symbols B (burning) and M (organ music). We use M, not O so it will not be confused with the digit zero. This will avoid a common error amongst students of the philosophy of logic.

Unfortunately the reply is torn at this point, presumably by a disapproving supernatural singer. Can you complete the solution for the good Mr. Cranston? He cannot reset the ghosts, so you will need a solution that works from any starting point. Also Mr. Cranston -really it is the marker - expects a state table with the states in K-map order. He will also need a sequence to follow for any starting point.