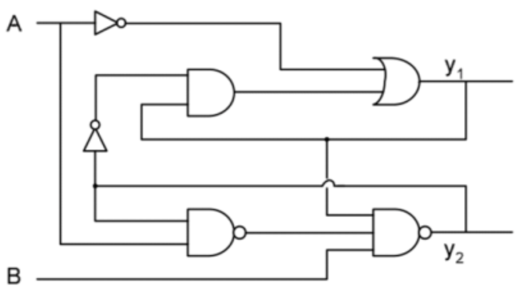
PS5

April 9, 2018 15:10

1. Draw a state diagram that illustrates the behaviour of the following asynchronous state machine:



Strategy : start with expression

y_1 next $= \bar{A} + y_1 \cdot \bar{y_2}$

y_2 next $= \overline{\left(B \cdot y_1 (\overline{y_2 \cdot A})\right)} = \bar{B} + \bar{y_1} + y_2 A$

y_1 next, y_2y_1 \ AB	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	1	1	0	0
10	1	1	0	0

y_2 next, y_2y_1 \ AB	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	1	1
10	1	1	1	1

→ state transitions

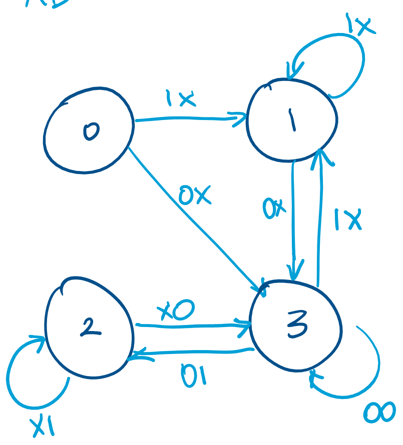
A	B	y_1	y_2	y_1'	y_2'
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	0	1

ideally swap these columns (easier to work with)

Let y_1y_2 be states : 00 → 0, 01 → 1, 10 → 2, 11 → 3

→ state transition diagram

input = AB



2. Consider an asynchronous state machine implemented using the following next-state equations:

Y1 = AB' + y1 y2 B

Y2 = AB + y1

Where A and B are inputs, B' means the inverse of B, and y1 and y2 are the current state wires.

Are there any static hazards (potential glitches) if these equations are implemented directly? If so, how can you eliminate the static hazard(s)?

$$\begin{array}{c|cccc} Y_1 \;\; y_1y_2 \backslash AB & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & 1 \\ 01 & 0 & 0 & 0 & 1 \\ 11 & 0 & 1 & 1 & 1 \\ 10 & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|cccc} Y_2 \;\; y_1y_2 \backslash AB & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 1 & 0 \\ 01 & 0 & 0 & 1 & 0 \\ 11 & 1 & 1 & 1 & 1 \\ 10 & 1 & 1 & 1 & 1 \end{array}$$

✓ Fix

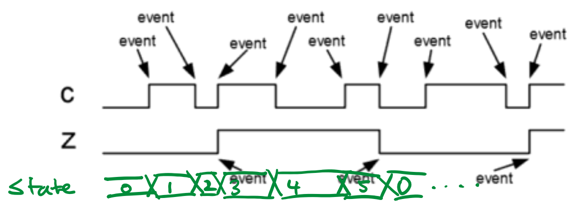
⚠ Hazard ⚠

✓

Y_1 has static hazards; can be fixed by adding the term such that equation B:

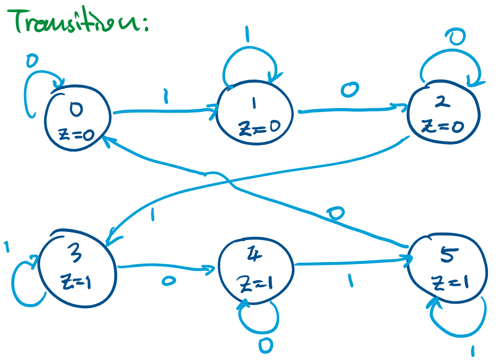
$$\boxed{Y_1 = A\bar{B} + y_1y_2B + y_1y_2A}$$

3. Consider an asynchronous state machine with one input **c** and one output **z.** The circuit produces an event on output **z** for every ***third*** event on input **c** (recall: event=toggle from 0 to 1 or 1 to 0). For example:



Design the circuit. You must use asynchronous design techniques. Your answer may not contain any flip-flops. Show your work clearly and state any assumptions. Present your answer in the form of ***either*** logic equations ***or*** a schematic (not Verilog).

State Transition:



	S_1	S_2	S_3
States: 0:	0	0	0
1:	0	0	1
2:	0	1	1
3:	0	1	0
4:	1	1	0
5:	1	0	0
	~~1~~	~~0~~	~~1~~
	~~1~~	~~1~~	~~1~~

S_1	S_2	S_3	C	S_1'	S_2'	S_3'	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	1
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	1	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

KMAP

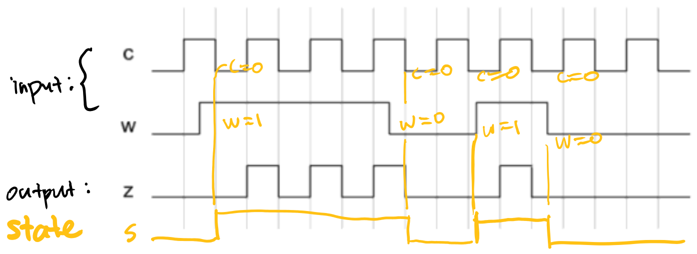
$$\begin{array}{c|cccc} S_1' \;\; S_1S_2 \backslash S_3C & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & 0 \\ 01 & 1 & 0 & 0 & 0 \\ 11 & 1 & 1 & X & X \\ 10 & 0 & 1 & X & X \end{array} \qquad \begin{array}{c|cccc} S_2' \;\; S_1S_2 \backslash S_3C & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & 1 \\ 01 & 1 & 1 & 1 & 1 \\ 11 & 1 & 0 & X & X \\ 10 & 0 & 0 & X & X \end{array}$$

$$\begin{array}{c|cccc} S_3' \;\; S_1S_2 \backslash S_3C & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 1 & 1 & 1 \\ 01 & 0 & 0 & 0 & 1 \\ 11 & 0 & 0 & X & X \\ 10 & 0 & 0 & X & X \end{array} \qquad \begin{array}{c|cccc} Z \;\; S_1S_2 \backslash S_3C & 00 & 01 & 11 & 10 \\ \hline 00 & 0 & 0 & 0 & 0 \\ 01 & 1 & 1 & 0 & 0 \\ 11 & 1 & 1 & X & X \\ 10 & 1 & 1 & X & X \end{array}$$

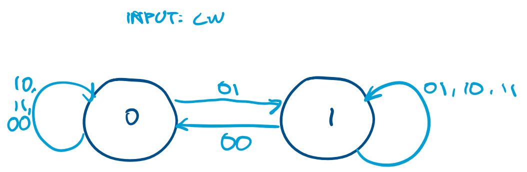
* prime in this case denotes the next value.

$$\boxed{\begin{aligned} S_1' &= \bar{S_3}\bar{C}S_2 + S_1S_2 + S_1C \\ S_2' &= \bar{S_1}S_2 + S_1\bar{S_2} + S_2\bar{C} \\ S_3' &= \bar{S_1}\bar{S_2}C + S_3\bar{C} + \bar{S_2}S_3 \end{aligned}} \qquad \boxed{Z = S_1 + S_2\bar{S_3}}$$

4. Design an asynchronous circuit that meets the following specifications. The circuit has two inputs: a clock input *c* and a control input *w*. The output, *z*, replicates the clock pulses when *w=1*, otherwise, *z=0*. The pulses appearing on *z* must be full pulses. Consequently, if *c=1* when *w* changes from 0 to 1, then the circuit will not produce a partial pulse on *z*, but will wait until the next clock pulse to generate *z=1*. If *c=1* when w changes from 1 to 0, then a full pulse must be generated; that is, *z=1* as long as *c=1*. The following diagram illustrates the operation of this circuit.



State Transition:



state	c	w	next state	out
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

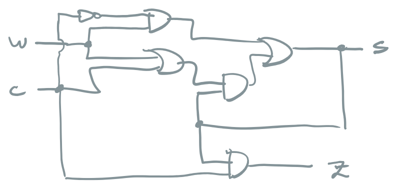
KMAP

s' \ cw	00	01	11	10
s 0	0	1	0	0
1	0	1	1	1

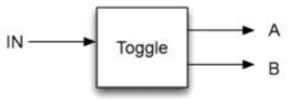
z \ cw	00	01	11	10
s 0	0	0	0	0
1	0	0	1	1

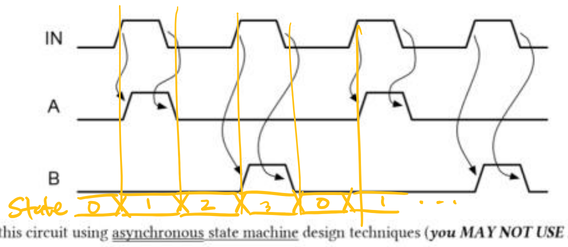
glitch masking for 1-bit state?

$$\boxed{\begin{aligned} S' &= S(W+C) + \bar{C}W \\ Z &= SC \end{aligned}}$$



6) Consider the following toggle circuit. The toggle circuit has a single input IN and two outputs A and B. Whenever IN is low, both outputs are low. The first time IN goes high, output A goes high. On the next rising transition of IN, output B goes high. On the third rising edge, output A goes high again. The circuit continues steering pulses on IN alternately between A and B.





Design this circuit using asynchronous state machine design techniques (***you MAY NOT USE A FLIP-FLOP***) .

State Trans.

state	IN	next	A	B
0	0	0	0	0
0	1	1	1	0
1	0	1	0	0
1	1	0	0	1

$$\left.\right\} \quad \begin{aligned} S' &= S \text{ XOR } IN \\ A &= \bar{S}(IN) \\ B &= S(IN) \end{aligned}$$

THIS IS WRONG!
→ will end up in undesired

$B = S(IN)$

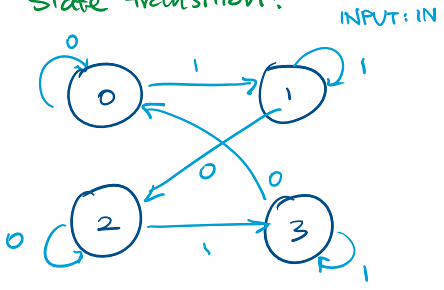
THIS IS WRONG!

$\Rightarrow$ will end up in undesired combiactional loop due to XOR

State encoding:

0:	00
1:	01
2:	11
3:	10

state transition:



State S_1	S_2	IN	next S_1'	S_2'	A	B
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
1	0	0	0	0	0	1
1	0	1	1	0	0	1
1	1	0	1	1	0	0
1	1	1	1	0	0	0

KMAP:

S_1'	S_2 IN 00	01	11	10
S_1 0	0	0	0	1
1	0	1	1	1

S_2'	S_2 IN 00	01	11	10
S_1 0	0	1	1	1
1	0	0	0	1

A	S_2 IN 00	01	11	10
S_1 0	0	0	1	1
1	0	0	0	0

B	S_2 IN 00	01	11	10
S_1 0	0	0	0	0
1	1	1	0	0

$$\boxed{\begin{aligned} S_1' &= S_1(IN + S_2) + S_2\overline{IN} \\ S_2' &= \overline{S_1}(IN + S_2) + S_2\overline{IN} \\ A &= \overline{S_1}S_2 \\ B &= S_1\overline{S_2} \end{aligned}}$$