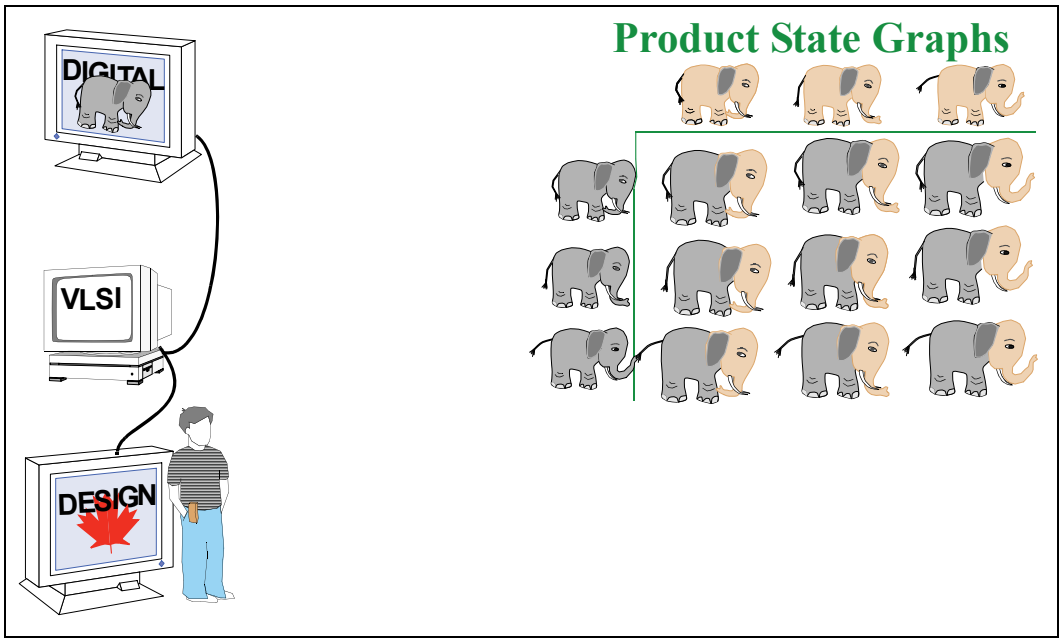


Carleton University 2008

■

Special State Graphs

Checking for Dual Sequences

1101 or 1011 Mealy Detection

Product Graph

010001 or 0101 Moore Detector

Product Graph

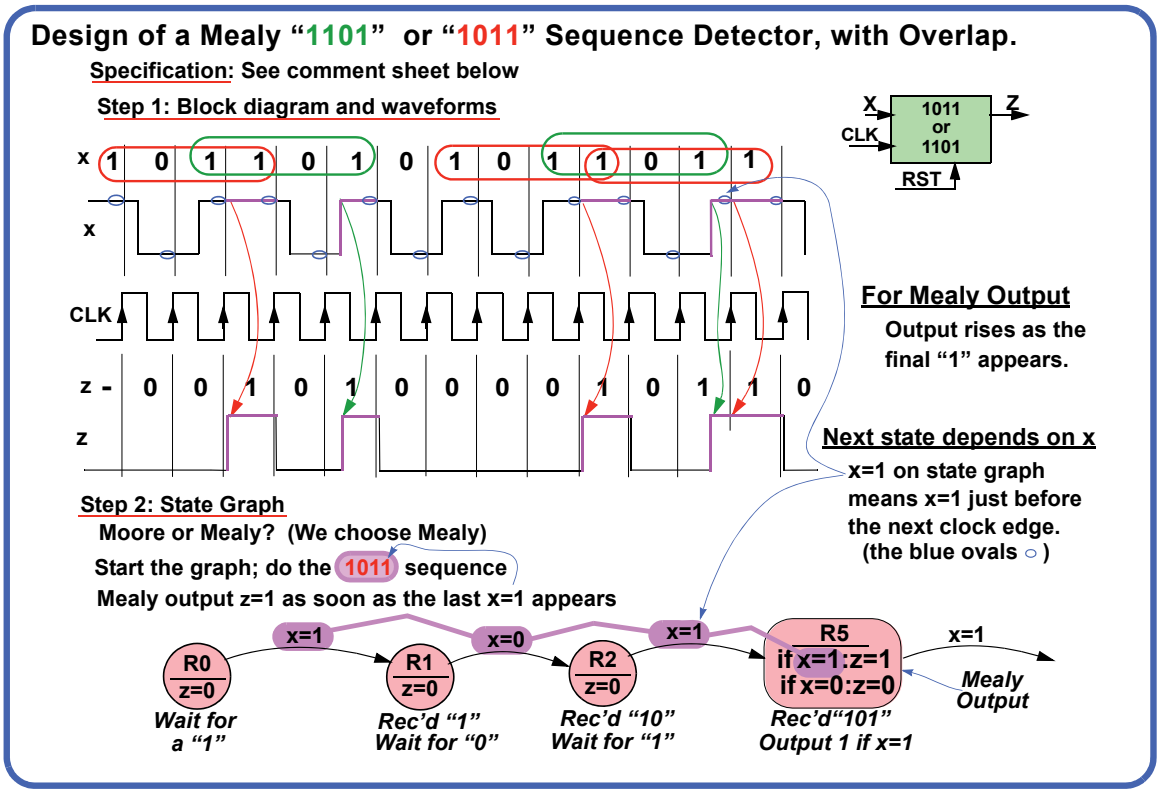
One-Pulse-Per-Push Circuit

Mealy

Moore

Coupled State Machines

Product Graphs: Example; A Dual Sequence Detector



1101 or 1011 Machine

Specification

This machine recognizes two sequences, 1101 and 1011. The sequences may overlap, so the end of one sequence may be the start of another. It gives a 1 output as soon as the final 1 is received.

Step 1:

Sketch a typical input with various nonoverlapping and overlapping sequences.

Sketch the Mealy output that rises as soon as the final 1 appears. It does not wait for the next clock edge. directly to one output with no fooling around.

The bit values like 0 0 1 ... given x and z, are the values near the end of the cycle.

Step2:

Draw the state graph.

We will draw the graph for one sequence. Then do the other sequence. Then combine them.

The Mealy outputs are written in the bottom of the state circles.

Remember that the next state is determined by x just before the active clock edge

The state changes after a clock edge. Only the Mealy output can change between clock edges.

The state names here were chosen to be short, but to also tell the sequence so far. Thus

R2 means "Received 2" which in binary is "Received 10"

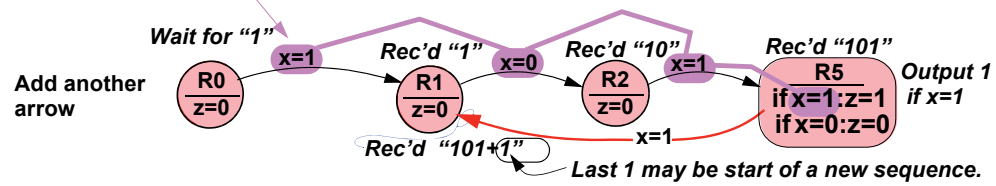
The state names G2 etc. will be used for the other sequence.

Product Graphs: The Top Sequence

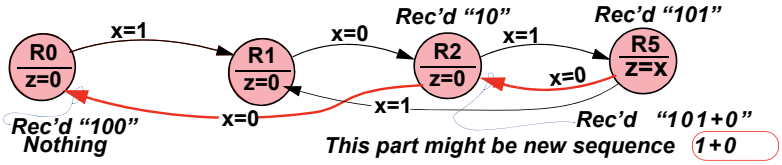
A Mealy "1101" or "1011" Sequence Detector with Overlap

Step 2: State Graph (Continued)

Continue the 1011 sequence alone; temporarily forget 1101



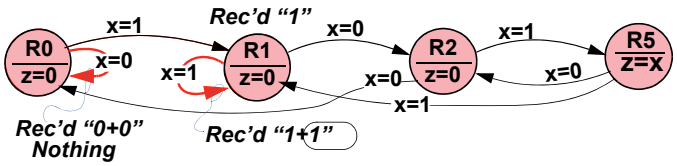
Add another arrow



Add two more arrows

Add last two arrows.

That's all, since now all states have 2 exit arrows.



1101 or 1011 machine

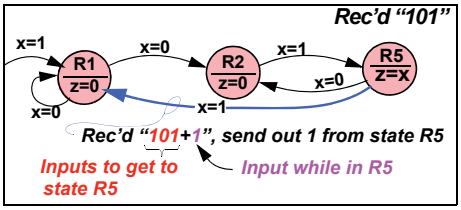
The 1011 sequence

There are two sequences to look for. We started by looking only for the 1011 sequence.

On the next page we will add more states to find the 1101 sequence.

a. Start by drawing the states for a 101 input.

b. Then continue trying to see what would happen for the last 1 of 1011.
The output notation z = x applies to the x after one reaches the state. Thus in state R5, one could think of the output changing according to the values of **x** on the arrows leaving the state. Only the blue x=1 arrow is associated an output from R5.



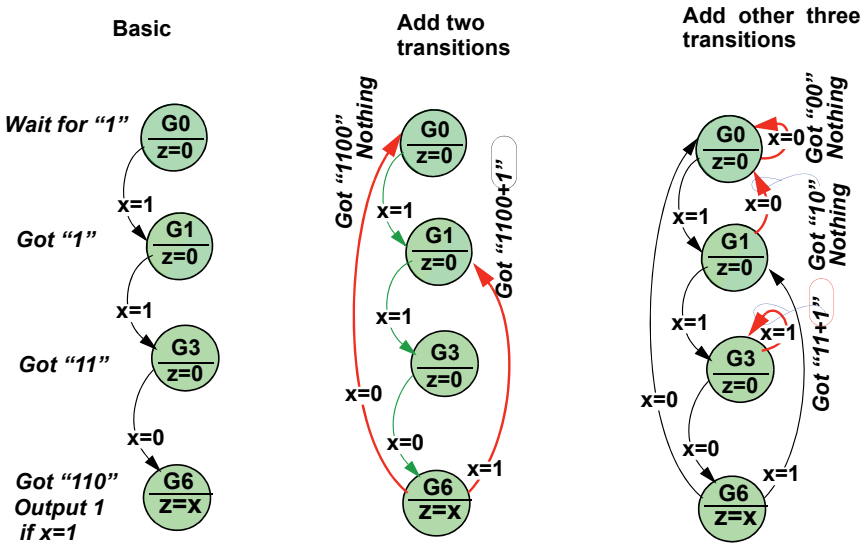
c. Since **x** can have two values, there must be two arrows leaving each state.
With two inputs there must be four arrows leaving each state.

Product Graphs: The Side Sequence

A Mealy "1101" or "1011" Sequence Detector with Overlap

Step 2: State Graph (Continued)

Do the 1101 sequence alone



The 1101 sequence

There are two sequences to look for. We started by looking only for the 1011 sequence.

On the next page we will add more states to find the 1101 sequence.

a. Start by drawing the states for a 110 input, which is the desired input.

b. Add the edge for the final 1 of 1101. The final 1 gives the output, but it also takes us to a next state with no ourput.rSince x can have two values

c. There must be two arrows leaving each state. Fill in all these arrows.

Product State Graphs: Combining Outputs

The Product State Graph for the Mealy "1101" or "1011" Sequence Detector

Step 2: State Graph (Combining the two graphs))

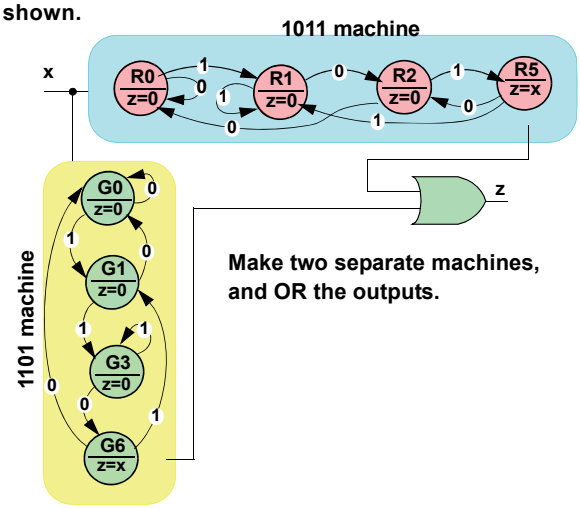
The two state graphs define the machine

You can make a a working machine as shown. It will be larger than needed.

The two graphs can be combined into a *product graph* (next slide)

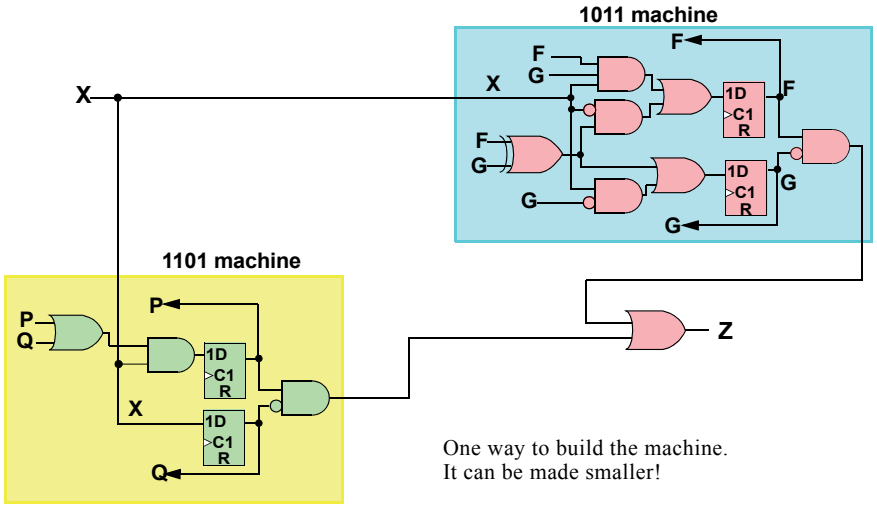
- **This has a large number of states.**
- **This *product graph* can be reduced to less than the two individual graphs.**
- **The reduced *product graph* usually gives a smaller circuit.**

START OF A "PRODUCT" GRAPH



Make two separate machines, and OR the outputs.

The Machine Made From Two Submachines.



One way to build the machine.
It can be made smaller!

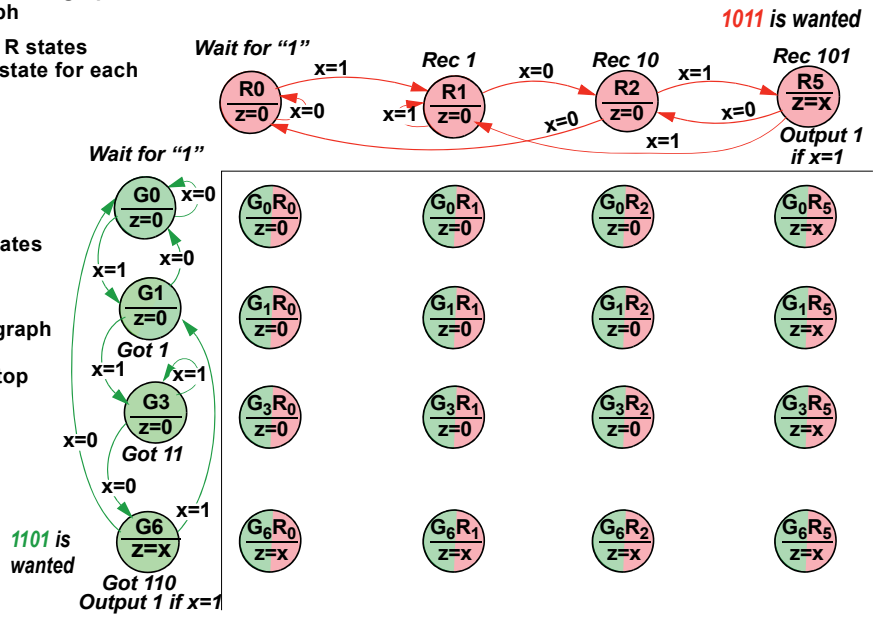
Product State Graphs: Combining States

The product states

A Mealy "1101" or "1011" Sequence Detector with Overlap

A new combined state graph
The product graph

Merge the G and R states to get a new GR state for each row and column.



Most of the new states will be unused.

The final product graph should be smaller than the side and top graphs together

One Way to Combine the Graphs

The combined graph will be done on the slides using a product graph.
Names
The original names like G5 means we got 101 (binary 5).

Here we give the state two names, thus G_3R_2 means we got 3 (binary 11) along the G (green) sequence, and 2 (binary 10) along the R (red) sequence, both simultaneously.

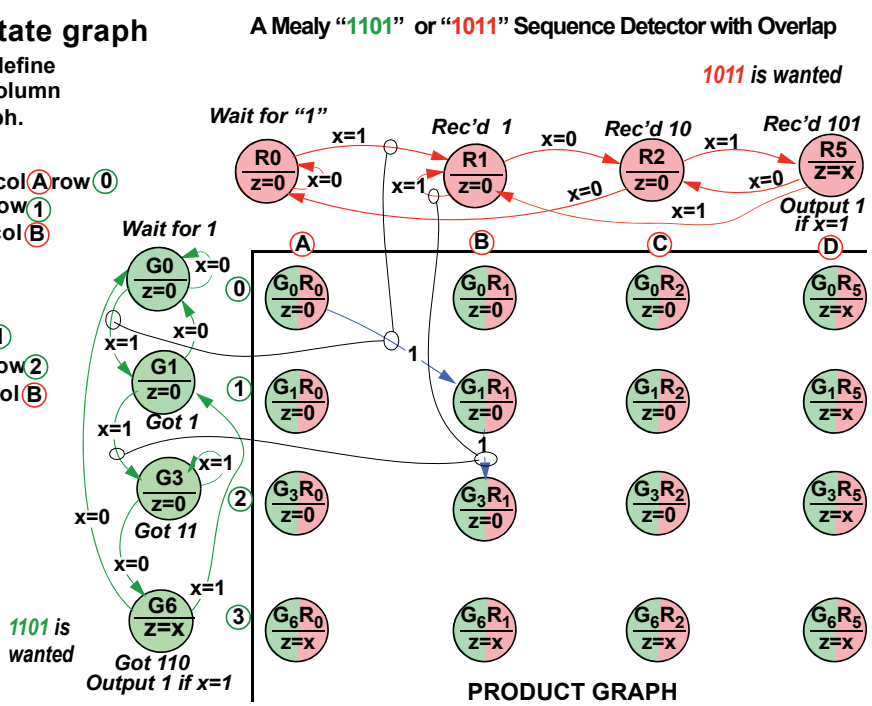
Product State Graphs: Transitions (Arrows)

The product state graph

The small graphs define the new row and column in the product graph.

Start in state G_0R_0 col A row 0
x=1 takes $G_0 \rightarrow G_1$ row 1
takes $R_0 \rightarrow R_1$ col B
thus $G_0R_0 \rightarrow G_1R_1$

In G_1R_1 col B row 1
x=0 takes $G_1 \rightarrow G_3$ row 2
takes $R_1 \rightarrow R_1$ col B
thus $G_1R_1 \rightarrow G_3R_1$



Product State Graphs: Transitions (Arrows) ■

Product State Graphs: Transitions (Arrows)

The product state graph (cont)

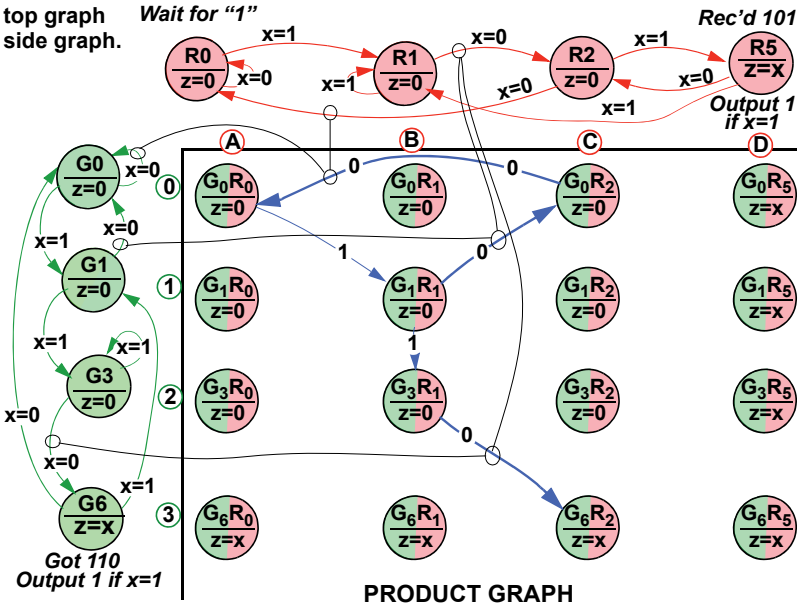
A Mealy "1101" or "1011" Sequence Detector with Overlap

Find where the arrows go in the product graph by finding:
1) the new column in the top graph
2) and the new row in the side graph.

Guides are shown for three arrows (transitions)

Some product states can never be reached. Transitions from these states are not needed.

Start new arrows only at states pointed to by a previous arrow.

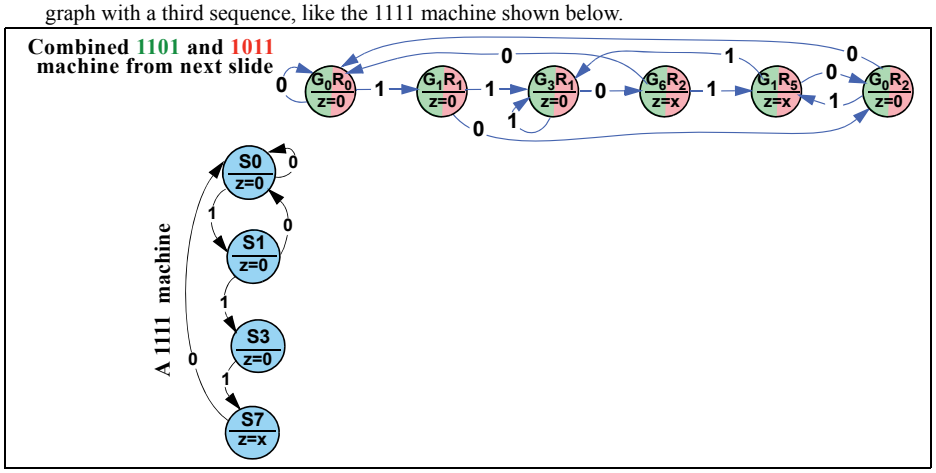


Product Graph

It is important to start from the starting (often Reset) state and work forward. If you start from a state you are not sure you can reach you may end up with extra useless arrows.

A Three-Sequence Graph

Detecting three simultaneous sequences would seem to lead to a 3-dimensional array which would get messy. However a brilliant student, Catalin Patulea, suggested that one take a completed 2-dimensional graph (like the one on the next slide) and straighten it so it becomes a 1-dimensional graph. Then one could make a product graph with a third sequence, like the 1111 machine shown below.

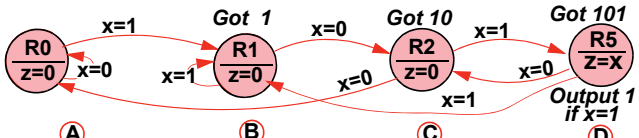


Product State Graphs: Transitions (Arrows)

Complete product state graph

Check all states with arrows entering them.
They must each have:
- An outgoing arrow for 1 input
- An outgoing arrow for 0 input

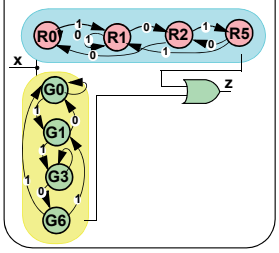
States with no entering arrows are unused.

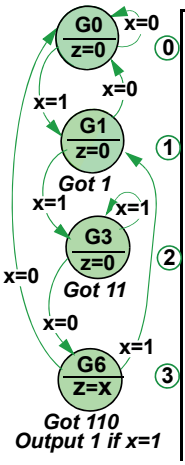


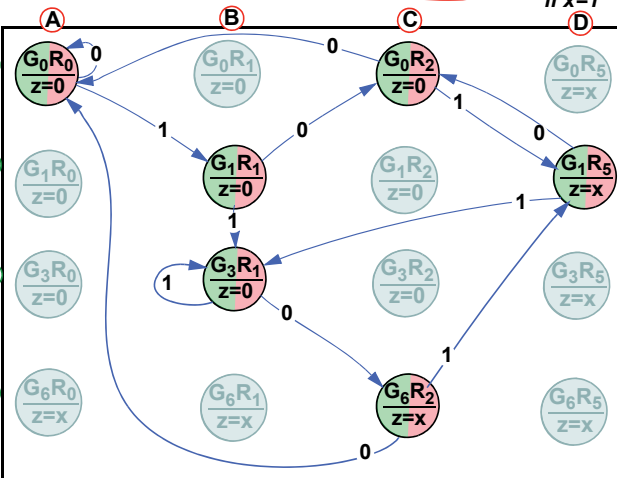
Compare

This Product Graph uses 6 states (3 FF)

The previous combined 1101 and 1011 graphs use 8 states (2 + 2 FF)





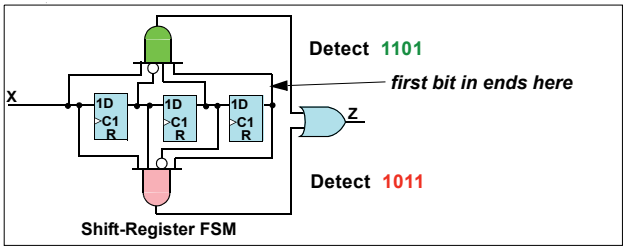


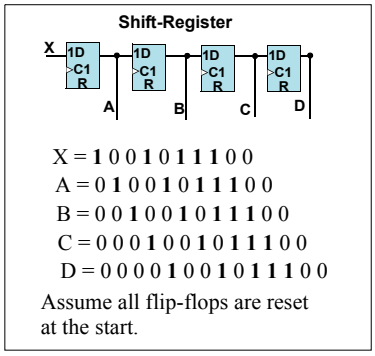
PRODUCT GRAPH

Shift Register Mealy Sequence Detector for 1101 and 1011 with Overlap

A Simple Solution

As an alternative to the previous design, a shift register can be used as an FSM to detect sequences. It works well where you need to identify several short sequences. It will not work for all FSMs.



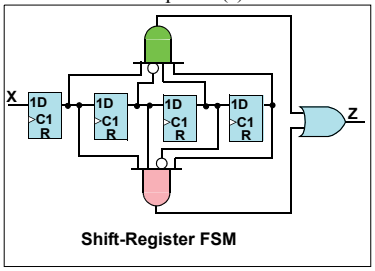


X = 1 0 0 1 0 1 1 1 0 0
A = 0 1 0 0 1 0 1 1 1 0 0
B = 0 0 1 0 0 1 0 1 1 1 0 0
C = 0 0 0 1 0 0 1 0 1 1 1 0 0
D = 0 0 0 0 1 0 0 1 0 1 1 1 0 0

Assume all flip-flops are reset at the start.

9-1.•PROBLEM

State what sequence(s) this circuit will find, and explain how it is different from the one above. Hint:Think M & M



Another Product State Graph Example

Synchronous State Graph 010001 or 0101 Detector

Design a state graph for a machine with:

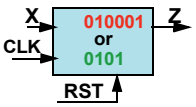
One input X, one output Z.

Z=1 after receiving the complete sequence 010001 or 0101

Overlapped sequences are detected.

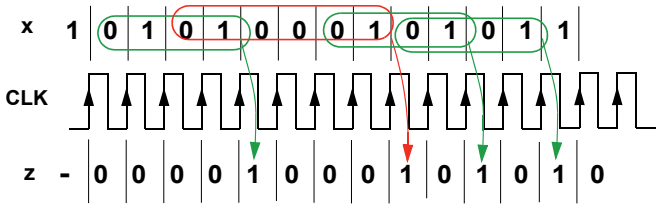
Z=0 except for the single clock period after the sequence is received.

Solution:



Step 1: Block diagram and waveforms

Moore Output: Output rises in clock period after sequence is received



010001 or 0101 Machine

This machine recognizes two sequences. The sequences may overlap, so the end of one sequence may be the start of another.

Step 1:

By now, you should be able to write the input sequence as 1s and 0s, knowing that these are the final values of the input as they appear just before the next active clock edge. These sequences should illustrate various overlapping sequences.

The output here is a Moore output, so it rises after the clock cycle in which the final 1 appears.

Step2:

Draw the state graph.

We will draw the graph for one sequence. Then do the other sequence. Then combine them.

The Moore outputs are written in the bottom of the state circles.

Remember that the next state is determined by x just before the active clock edge

The state changes after a clock edge. Also Moore output only change just after the active clock edge.

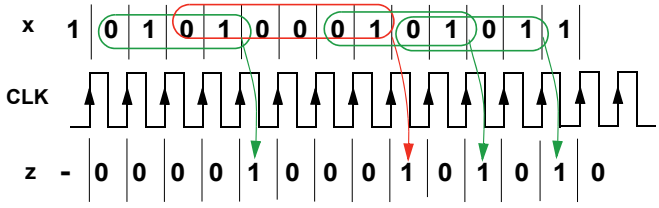
9-2.•PROBLEM

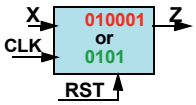
Design the state graph for a Mealy machine with overlap, which detects the seqence

Product State Graph; A Larger Machine

Design of a Moore "0101" or "010001" Sequence Detector, with Overlap.

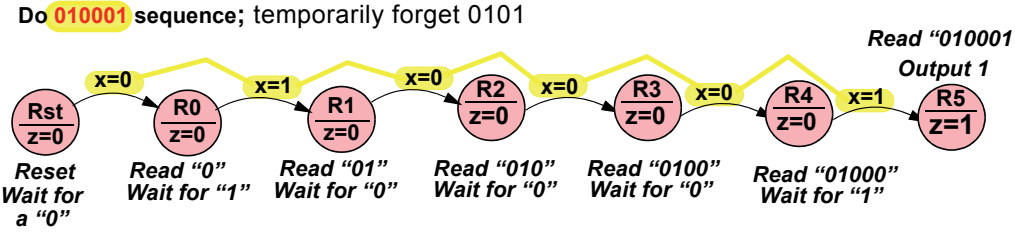
Step 1: Block diagram and waveforms





Step 2: State Graph

Do 010001 sequence; temporarily forget 0101



1101 or 1011 machine

The 1011 sequence

There are two sequences to look for. We started by looking only for the 010001 sequence.

Two pages later we will consider a machine to find the 0101 sequence.
Then we will combine them as a product machine.

d. Start by drawing the sequence of to capture the 010001 input.

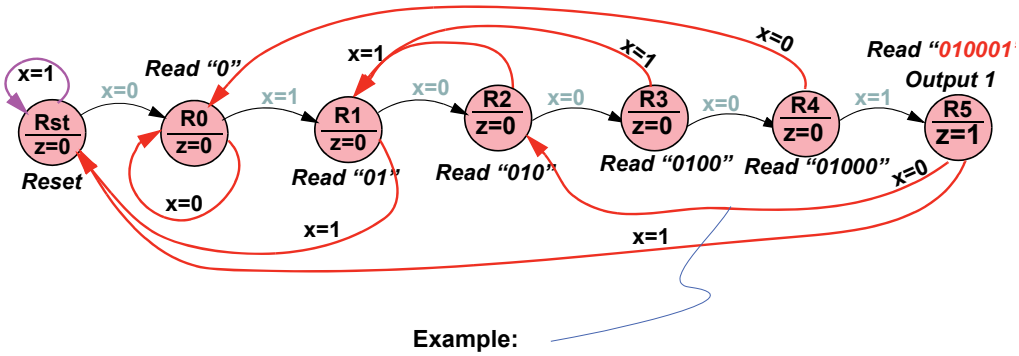
e. Then next page will add on what happens when the sequence is not this sequence.

Product State Graph; The Horizontal Sequence

A Moore "0101" or "010001" Sequence Detector with Overlap

Step 2: State Graph Continued

Complete the 010001 sequence



Example:
After we Read 010001 where next?
Let x=0.
Tack the 0 on the end "0100010
Is the end part of the desired sequence?
Yes, 010 is the start of a new sequence and would have taken us to R2.
Send the next state arrow to R2.

The 010001 sequence

There are two sequences to look for. We started by looking only for the 01001 sequence.

On the next page we will add more states to find the 1101 sequence.

Here we noted that with one input each state must have two exit arrows, one for a 1 input and one for a 0 input.

Even if the input does not come as 010001 directly, one may be able to find the sequence later. Thus from each state, one has to check the previous input bits, and check for 0, 01, 010, 0100, to see if one is part way through the desired sequence.

There must be two arrows leaving each state. Fill in all these arrows.

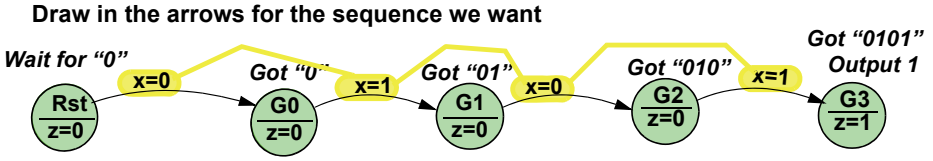
Product State Graph; The Vertical Sequence

A Moore "0101" or "010001" Sequence Detector with Overlap

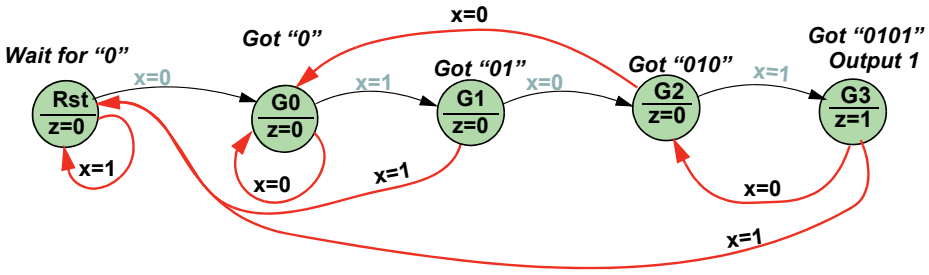
Step 2: State Graph (Continued)

Do the 0101 sequence alone; temporarily forget 010001

Draw in the arrows for the sequence we want



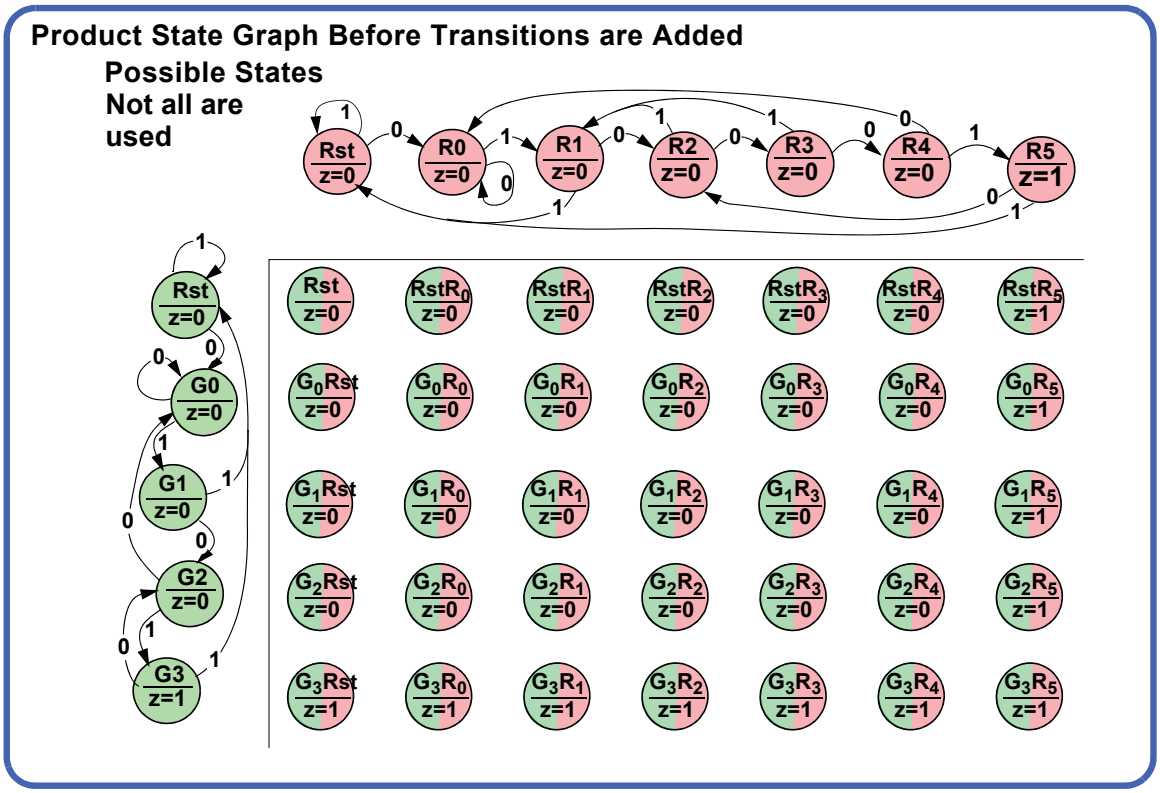
Draw in the side arrows to give to arrows out of each state



The 0101 Submachine

We now have two machines, one two detect each sequence. The next step is to consider both machines operating at once.

Product State Graph: Raw States



One Way to Combine the Graphs, the Product State Graph

The combined graph will be done on the slides using a product graph.

Since there are two machines, one is always in one state say the G states) and the other is in another (say the R states). When both machines are considered together, this whole combine machine is in a combined state made from the two other states.

Consider a machine four R states, which requires 2 flip flops to store the state. Also suppose the G machine has four states and it uses another two. The combined machine has 4 flip flops.

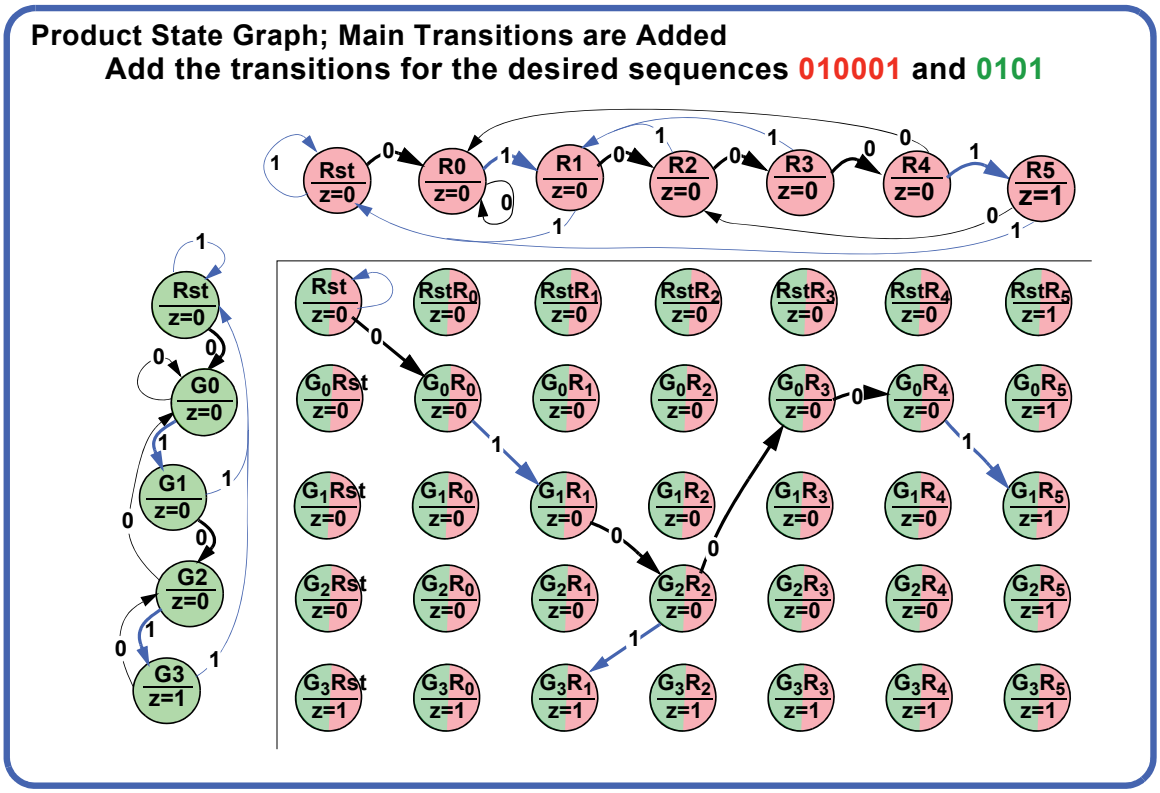
Thus if the R machine is in state 11, and the G machine is in state 00, the combined machine would be in state 1100 (or 0011 if you wanted to order the flip flops differently).

One can see the combined machine would have 4x4 = 18 states.

Here the R machine has 7 states and the G machine has 5, so the product machine has 5x7 = 35 states.

However they frequently are not all used.

Product State Graph; The Main Sequences



The Product State-Table

Having named the stages, and plotted them on an x-y grid, the next step is to add transitions.

Start by adding the transitions that would happen if the machine received exactly the right sequence.

First 0101 and then 010001.

Starting at Rst

A 0 is received:
Look at the green machine, and one sees a 0 takes one to G0. i.e. row 2.
Look at the red machine, and one sees a 0 takes one to R0, i.e. column 2.
Hence the next product state is G_0R_0 in row 2 and column 2.

Now let a 1 be received.

For the green machine, starting in state G0, a 1 takes one to state G1
For the red machine, starting in state R0, a 1 takes one to state R1.
Hence for the product machine, starting in state G_0R_0, a 1 takes one to state G_1R_1.

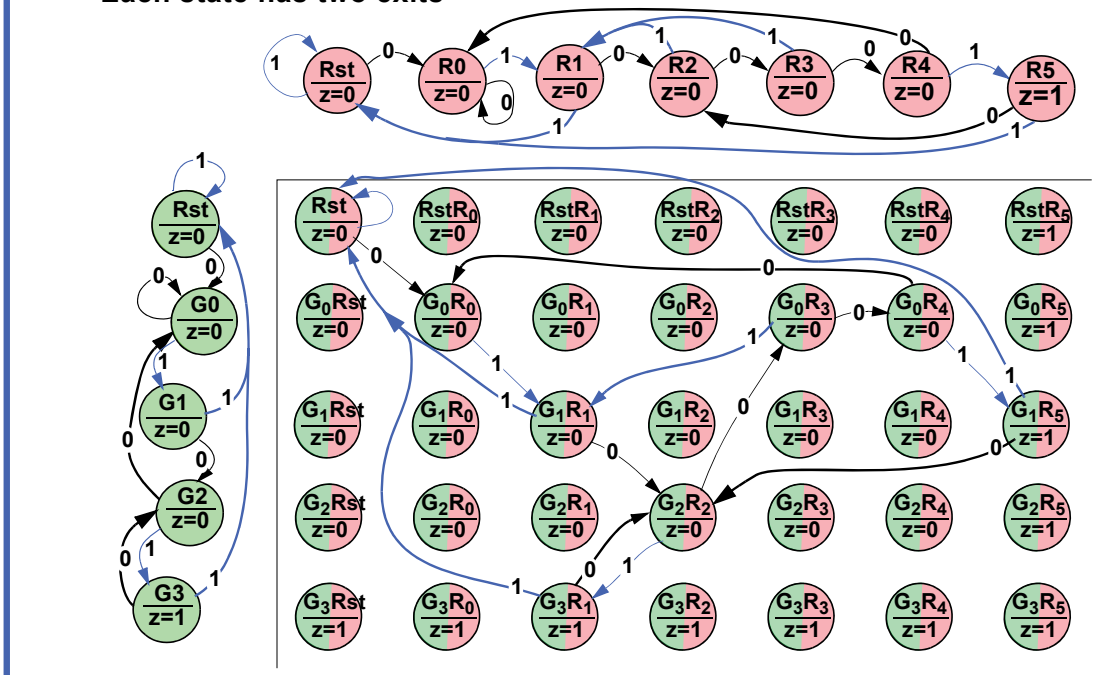
Continue adding the transitions (that's jargon for the arrows) for the 0101 sequence.
Then repeat for the 010001 sequence starting from Rst.

The Product State-Table

One could, instead of adding the transitions to the graph, make a state table directly. In this case it turns out to be more work. In some cases, particularly where the graph is large and messy, the state table is easier.

Product State Graph; Other Sequences

Product State Graph; Add the transitions leaving states reached before.
Each state has two exits



Transitions in the Product State Table

Filling in Transitions not done before

Several of the states that were reached on the last slide, have only one, or perhaps zero, exit arrows. Each state needs two exit arrows, so fill those in now.

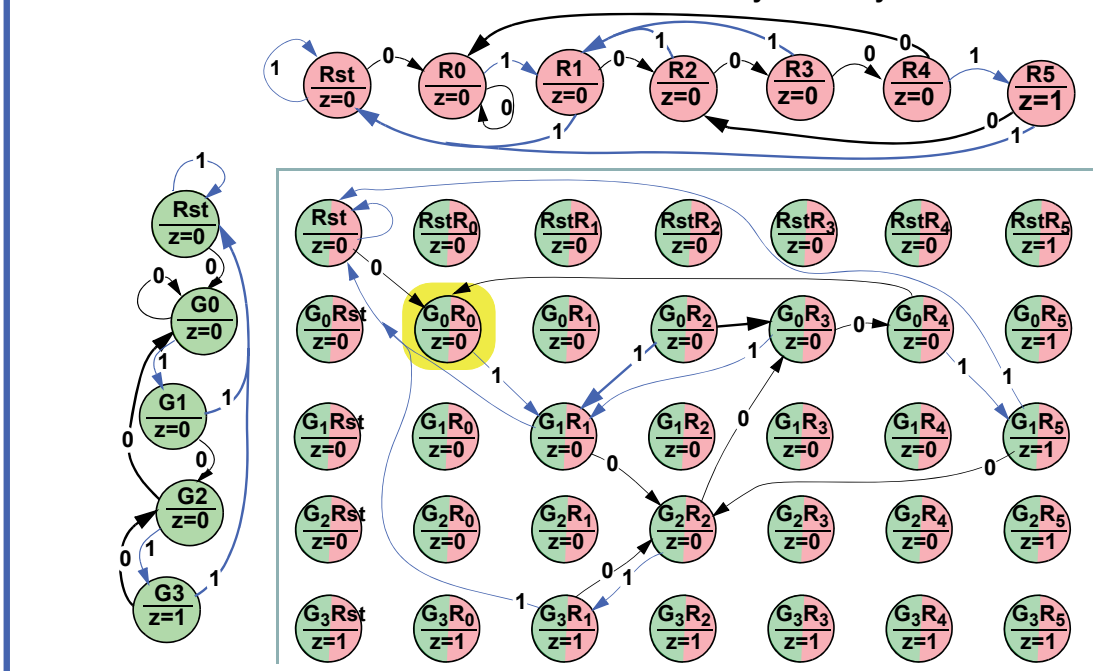
Note that one state G_1R_2 now has an arrow going into it, whereas previously it did not.

We will add the exit arrow from G_1R_2 on the next page.

Product Graph; All States Must Have Two Exit Arrows

Product State-Graph

Recheck that each state has two exits. Are there any with only one?

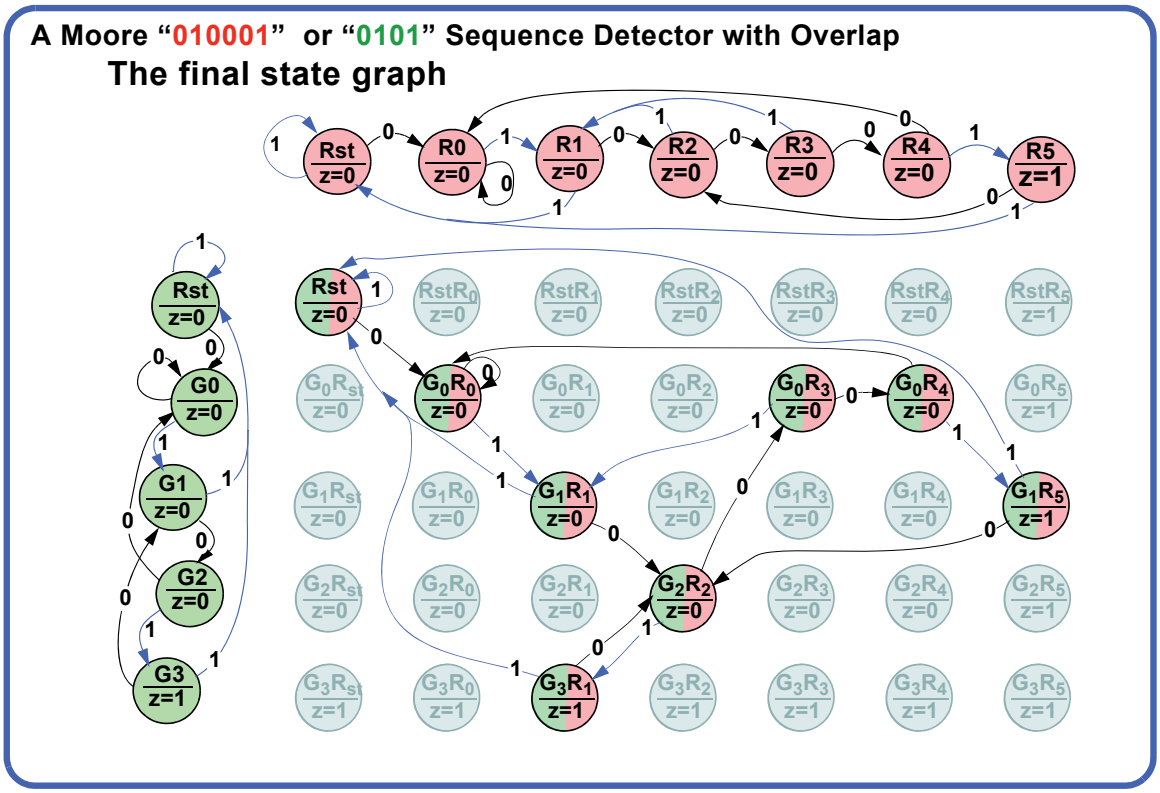


The Product State Table

Check again that all the states that can be reached have two exit arrows. If so, check the arrows, and if there were no errors, the graph is complete.

Fix the omitted arrow above.

The Final Product State Graph



The Final Product State Graph.

Comments

Here the final state graph has **eight** states requiring 3 flip-flops

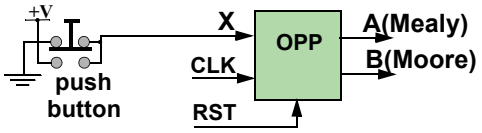
If one had used two separate machines and ORed their outputs, this would have worked, but it would have required 3 flip flops for the **five** state machine, and 3 more for the **seven** state machine for a total of 6 flip-flops.

Also, if a shift register machine had been used, it would have had to have a flip-flop for each bit of the longest input sequence, or 6 flip-flops.

A One Pulse Per Push Circuit; Mealy Design

Design of FSM from Word Description

A One Pulse-Per-Push Circuit (OPP)



Specification

Input X

A pulse which:
- always overlaps at least one clock edge
- may last many clock cycles.

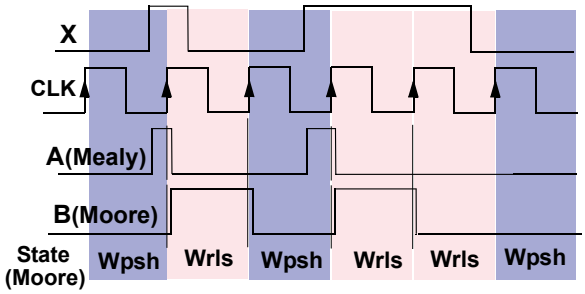
Output

Mealy output A

An output which:
- starts when the input starts
- finishes at the next clock edge.

Moore output B

An output which:
- starts at the next clock edge
- finishes one clock cycle later.



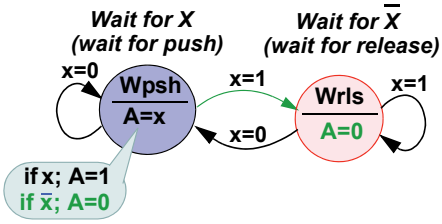
Machine With Mealy Output

Step 1

Block diagram, waveforms.

Step 2

State graph
- Add state to waveforms
- State changes only at clock edge
- Mealy output may change when x changes



Design of OPP[1] Machine

At the Start of the Design, Choose Whether You Want Mealy or Moore Outputs

This was discussed near the end of the last section. One important reason for this circuit is given below.

Duration of output

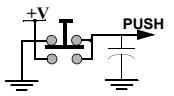
The Moore output always lasts a full clock cycle.

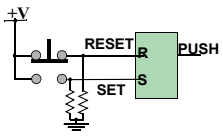
The Mealy output can be much less than a clock cycle. Its length depends on how far along in the cycle the input X changed. If the input comes late, the Mealy output can be very short, maybe too short.[2]

Input Push Buttons

The push button is connected so pushing it sends out a "1" signal. Here we assume the signal does not bounce. This might be true if a capacitor is connected to hold the last voltage value when for the short time the button is in between the contacts.

A more reliable debounce circuit was give in the Section on Latches and Flip Flops. and is repeated on the right.





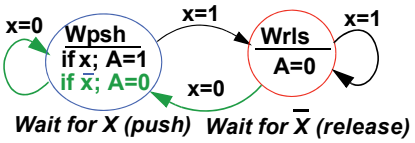
1. In Ontario Canada, OPP stands for Ontario Provincial Police. There is no connection between them and this circuit.

2. In real circuits inputs are sent through a single flip-flop before being allowed to enter the OPP circuit. This is discussed in the next course, ELEC 3500, under the topic "Asynchronous Inputs to Synchronous Circuits."

The One-Pulse-Per-Push Circuit; Mealy Circuit

Design of the Mealy FSM

A Mealy One Pulse-Per-Push Circuit



Step 3; State Table

Symbolic State Table

State Q	Nxt State Q^+ x=0	Nxt State Q^+ x=1	output A x=0	output A x=1
Wpsh	Wpsh	Wrls	0	1
Wrls	Wpsh	Wrls	0	0

State Table,

State Q	Nxt State Q^+ x=0	Nxt State Q^+ x=1	D input x=0	D input x=1	output A x=0	output A x=1
Wpsh=0	Wpsh=0	Wrls=1	0	1	0	1
Wrls = 1	Wpsh=0	Wrls=1	0	1	0	0

Step 4

State Assignment
Let Wpsh=0, Wrls=1

Step 5

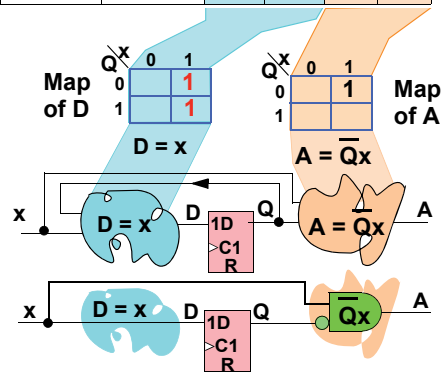
Put Assignment in State Table
Make K Maps

Step 6

Derive equations

Step 7

Draw Circuit



Design of Mealy OPP Machine

Symbolic State Table

This has only letters (symbols) for the states.

State Table (Complete)

This is made in Step 5 after the assignment of a bit pattern to each symbol. Here the bit pattern is just a single bit, since there are only two states.

The One-Pulse-Per-Push Circuit; Moore Design

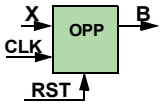
Design Moore FSM from Word Description

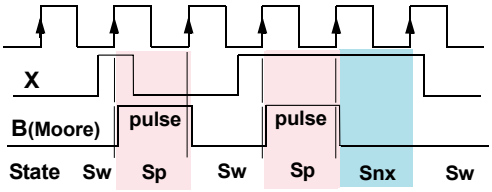
A Moore One Pulse-Per-Push Circuit

Step 1

Block diagram.

Draw Moore waveforms.

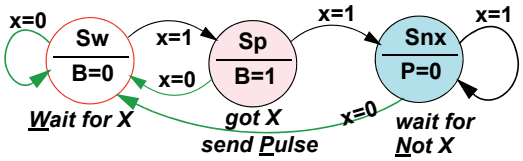




Step 2

State graph

- Add state to waveforms
- State changes only after clock edge



Step 3; State Table

Symbolic State Table

State Q	Nxt State Q^+ x=0	Nxt State Q^+ x=1	output B
Sw	Sw	Sp	0
Sp	Sw	Snx	1
Snx	Sw	Snx	0

State Table

State Q	Nxt State Q^+ x=0	Nxt State Q^+ x=1	output B
Sw=00	Sw=00	Sp=11	0
Sp=11	Sw=00	Snx=01	1
Snx=01	Sw=00	Snx=01	0

Step 4

State Assignment

Let Sw=00, Sp=11, Snx=01

Step 5

Put Assignment in State Table

Make K Maps

Must change to K-map order

Moore OPP

Compare Moore With Mealy

- Moore has an extra state
- The extra state usually means slightly more hardware.
  Going from two states to three requires another flip-flop.
- Moore output is delayed till the next clock period after the Mealy output.
- Moore outputs are close to a full clock cycle wide.[1]

[1] If there is a lot of logic in the output circuitry, this may be reduced, but it will be wider than a similar Mealy output.

The One-Pulse-Per-Push Circuit; Moore Circuit

A Moore One Pulse-Per-Push Circuit

Step 5

Put state table in K-map order

Make K Maps

Step 6

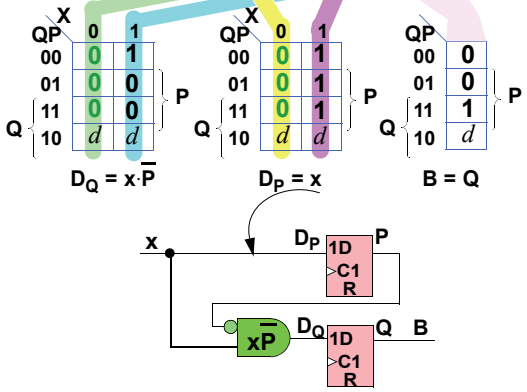
Derive equations

Step 7

Draw circuit

State Table

State Q,P	Nxt St $Q^+P^+=D_QD_P$ x=0	x=1	output B
Sw=00	Sw=00	Sp=11	0
Sp=01	Sw=00	Sn=01	0
Sn=11	Sw=00	Sn=01	1
	d d	*d d*	*d*



Moore OPP

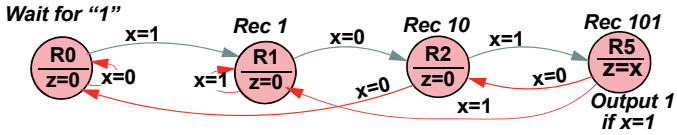
Notice we used an extra flip-flop. Going from one to two looks like a large increase. However when a machine has 8 flip-flops, going from Mealy to Moore would at most increase the number of flip-flops by one.

Summary and Common Errors

Making Product Graphs

1. **Check that each state has two outputs (for a one input machine).**

2. **The most mistakes are made when calculating where to go when off the main sequence.**



The red lines are the hardest

3. **Don't look at states that cannot be reached from RESET.**

When constructing a state graph, one should check for extra states.
This is done in the next chapter.

This example for the 1011 1101 machine done here close to the design of the machine.

Check for Extra States in Circuit of Slide 8

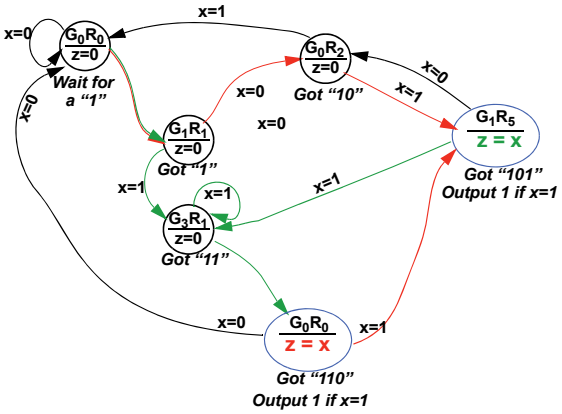
Checking the 1011 1101 State Graph tor Unneeded States

Don't read this until you read the chapter on state merging.

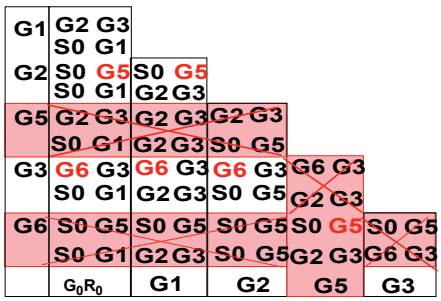
Checking for possible state mergers.

Some states may be redundant

Normally done for a FSM design.



State	Next State		Outputs	
S0	S0	G1	0	0
G1	G2	G3	0	0
G2	S0	G5	0	0
G5	G2	G3	0	1
S4	G6	G3	0	0
G6	S0	G5	0	1



No Mergers Possible

Something From the Next Chapter (cont)