EDA322: Digital Design
Exam - August 2017

Date: August 24, 2017

Time: **14:00-18:00**

Examiner: Ioannis Sourdis

Department: Computer Science and Engineering

Inquiries: Ioannis Sourdis (extension 1744); will visit the room at **15:30** and at **17:00**

Results and grading review: room 4128 EDIT on **September 15th at 11:00**.

Duration: 4 hours

Grading scale: 100 points in total

Chalmers:
0: 0%-49%, 3: 50%-64%, 4: 65%-84%, 5: 85%-100%
GU:
Fail (U): 0%-49%, Pass (G): 50%-79%, Pass with Distinction (VG): 80%-100%

Available references: a calculator is allowed. No textbooks or lecture notes, etc. allowed.

General: Submit your solutions, in English, on blank paper sheets. Write legibly; feel free to use figures to get your point across.

The order of answering the questions does not matter (start with the easiest ones).

Please start the solutions for each problem on a new sheet. Please number the sheets so that the solutions are in numerical order.

Note that it is possible to receive partial credit for an answer even if it is not 100% correct.

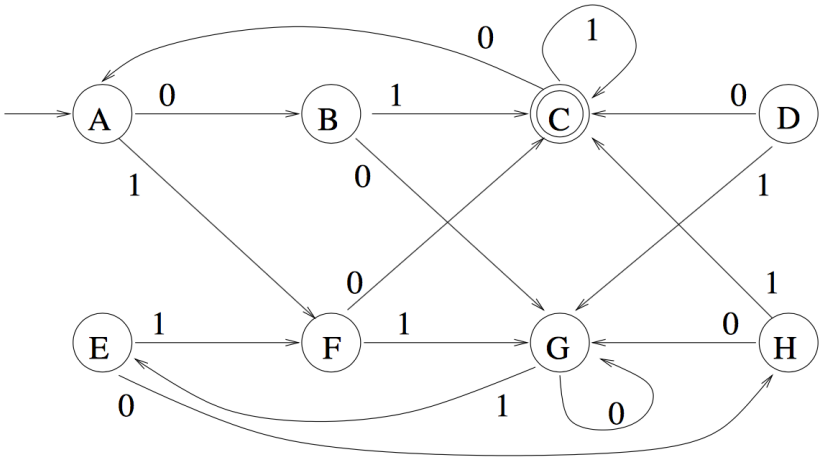
Your personal identity code is required on each submitted sheet!

Good luck!

Question 1: (10 points)
Design in gatelevel a circuit that takes as an input a 4-bit integer A(3:0) and produces the integer part of the result of the following equation R= A*(9/8)

Answer:
R= A*(9/8) = A + A/8
A/8 requires a shift right of 3 bits and then the result should be added to A with a 4-bit adder

Question 2: (10 points)
Minimize the states of the FSM described by the following state diagram using an implication table, then, rewrite the state diagram:



All states give "0" (zero) to the output except state C which gives "1".

Answer:

Step 1: We have one unreachable state, state D. We remove this state and proceed to Step 2.

Step 2: We first mark pairs of final and non-final states, getting the following tableau:

B						
C	X	X				
E			X			
F			X			
G			X			
H			X			
	A	B	C	E	F	G

In the first iteration we examine all unmarked pairs. For example, for pair A,B, we get δ(A,1) = F and δ(B, 1) = C, and the pair C, F is marked, so we mark A, B too. After doing it for all pairs, we get the following tableau.

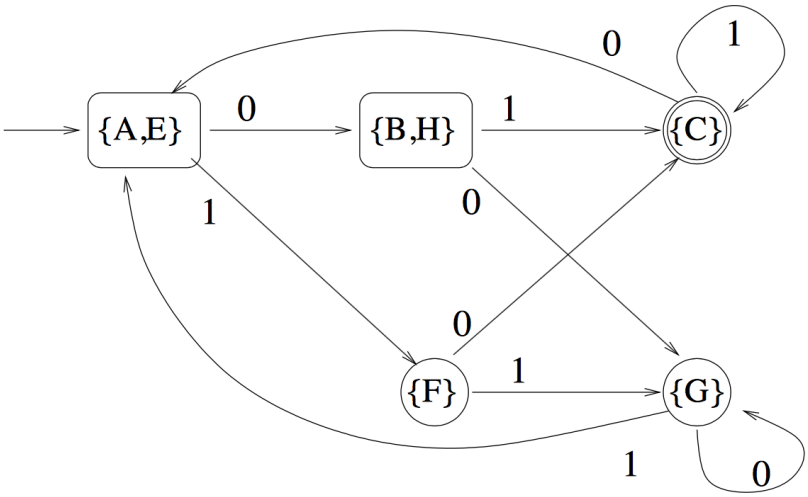
B	X					
C	X	X				
E		X	X			
F	X	X	X	X		
G		X	X		X	
H	X		X	X	X	X
	A	B	C	E	F	G

In the next iteration, we examine the remaining pairs. For example, we will mark pair A,G because δ(A, 0) = B and δ(G, 0) = G, and the pair B, G is marked. When we're done we get the following tableau:

B	X					
C	X	X				
E		X	X			
F	X	X	X	X		
G	X	X	X	X	X	
H	X		X	X	X	X
	A	B	C	E	F	G

One more iteration will be executed now, but no new distinguishable pairs will be discovered. So we are done with Step 2 now.

Step 3: We group the states into the equivalence classes. Since A, E are equivalent and B, H are equivalent, the classes are: {A, E}, {B, H}, {C}, {F }, {G}. The minimal automaton Aˆ is:



Question 3: (10 points)
a) What is the main (types of) "resources" that an FPGA contains? What other specialized modules may be included in an FPGA?
b) What are the most area and power consuming resources? Why is that in your opinion?

Answer:
Lecture 11, slides 9, 35-39, 42, 44
(The mentioned lecture slides can be found at the end of this document)

Question 4: (10 points)
Draw the block diagram of a bus that connects 4 clients and the block diagram of a crossbar that connects 4 clients. Consider that a client makes a transaction (communicates data to another client) when the bus or the crossbar allows with a delay of one cycle. Show the timing of the following transactions (all requests are created and are available in cycle 0):

- Client 1 to Client 2
- Client 2 to Client 1
- Client 3 to Client 4
- Client 4 to Client 3
- Client 1 to Client 4
- Client 1 to Client 2
- Client 2 to Client 1

How many cycles does a bus need for the transactions and how many does the crossbar?

Answer:
Crossbar needs 3 cycles, bus needs 7

Question 5: (10 points)
How can a memory block be used to implement combinational logic? When in your opinion would that be useful? What are the 3 types of Programmable Logic Devices and what are their main differences?

Answer:
Lecture on interconnects and memory slide 37.

It would be useful if the combinational logic is too complex to implement with gates or it is required to be dynamically changed. In the second case the memory could be reprogrammed to like an FPGA logic cell to implement a new Boolean function.

Lecture on interconnects and memory, slide 39

Question 6: (10 points)
a) What is the difference between testing and verification?
b) What are the 3 different types of faults that can happen in a digital circuit and what are their causes?

Answer:
Lecture on Testing, slides 5, 11

Question 7: (10 points)
a) Draw the block diagram of a 4-bit array multiplier, showing the internals of a full-adder at least once.
b) Show the critical path of the design considering that the inputs and outputs of the multiplier are registered.
c) Considering that the delay of a XOR gate is 0.5 ns, the delay of an AND/OR gate is 0.2 ns, NOT gates have zero delay, the setup time of a flip-flop is 0.1 ns, the hold time of a flip-flop is 0.01ns and the propagation time of a flip-flop is 0.1 ns. Find the minimum clock period of the design.
d) Show how to best split the design in **two** pipeline stages in order to achieve the highest frequency. Show/Mark the wires in the block diagram that would be registered in this case. What would then be the minimum clock period of the design.

Answer:
a) Lecture on Arithmetic Units, slide 17-18 (plus showing the FA internals)
b) Lecture on Arithmetic Units, slide 18
c) 200 MHz (clock period 5 ns) 23-27, 32-18
d) All wires crossed by X3 and its projection to the right would be registered, then the critical path would be 2.7ns and the operating frequency 370 ns.

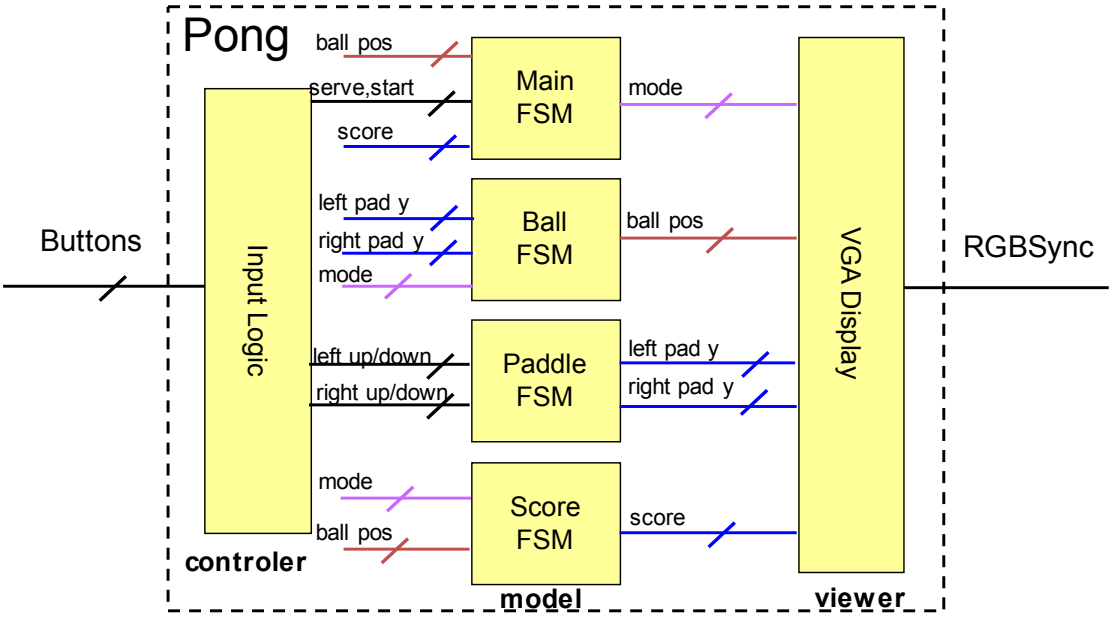
Question 8: (10 points)
Draw the block diagram and truth table of an SR-latch. Analyze the circuit following the methodology used in the asynchronous circuits and explain what would happen when if inputs change from SR=11 to SR=00

Answer:
Lecture on Asynchronous Sequential Logic, slides 18-20

Question 9: (10 points)
a) What does the delay of a gate depend on?
b) What does the delay of a wire depend on?

Answer:
Lecture on Timing, Delay, Power, slides 9-17

Question 10: (10 points)
After defining the specifications of a system to be designed, a common practice to proceed with the designing step is to follow a Divide and Conquer approach.
a) Explain what is the Divide and Conquer approach in that context.
b) Describe the 3 different themes (styles/ways) you can apply Divide and Conquer for the design of a system.
c) What is (are) the theme(s) based on which the following system is designed. Explain why.

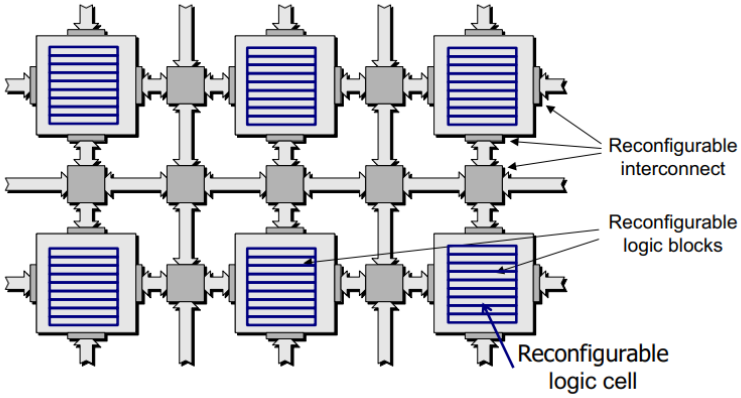


Answer:
Lecture on System Design and Interfaces, slides 5-9

END of EXAM

Question 3:

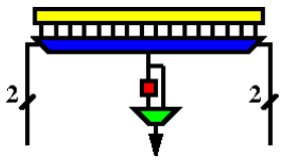
FPGA devices

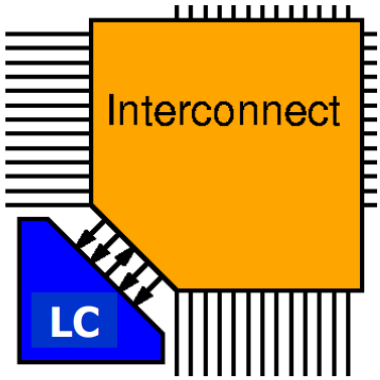


EDA322 Digital design, 2017-2018, Lecture 11 I. Sourdis, CSE, Chalmers 9

Logic Cells: Conventional FPGA Tile

K-LUT (typical k=4) w/ optional output Flip-Flop





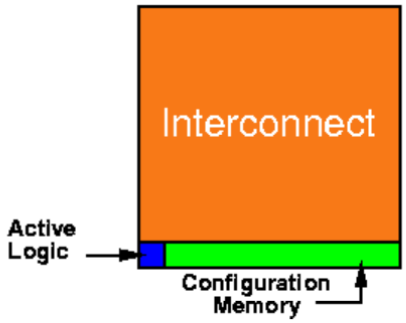
EDA322 Digital design, 2017-2018, Lecture 11 I. Sourdis, CSE, Chalmers 16

Specialized Modules:

Heterogeneous reconfigurable environments

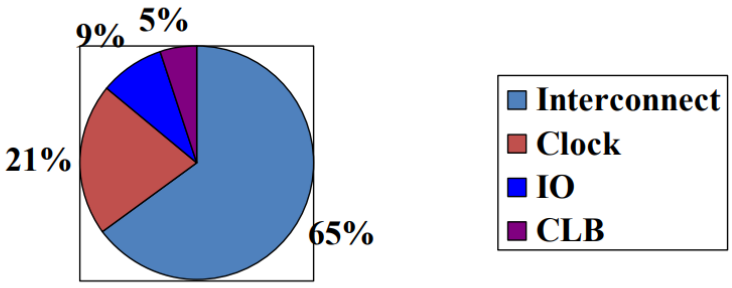
- Reconfigurable fabric might contain non-reconfigurable elements that interface to the logic blocks through the reconfigurable interconnect fabric
- Examples:
  - Embedded memory
  - Embedded multipliers, adders, MAC
  - Embedded processors (e.g. PPC)

Area



Function	Area (λ^2)
LUT MUX + ff	20K (generous, closer to 10K)
Programming Memory	80K (240K typical unencoded)
Interconnect	700K (for $N_p = 2048$)

Power

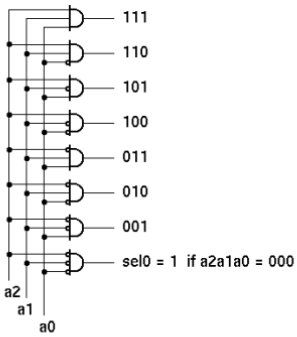


XC4003A data from Eric Kusse (UCB MS 1997)

Question 5:

Relation between Memory & Combinational logic

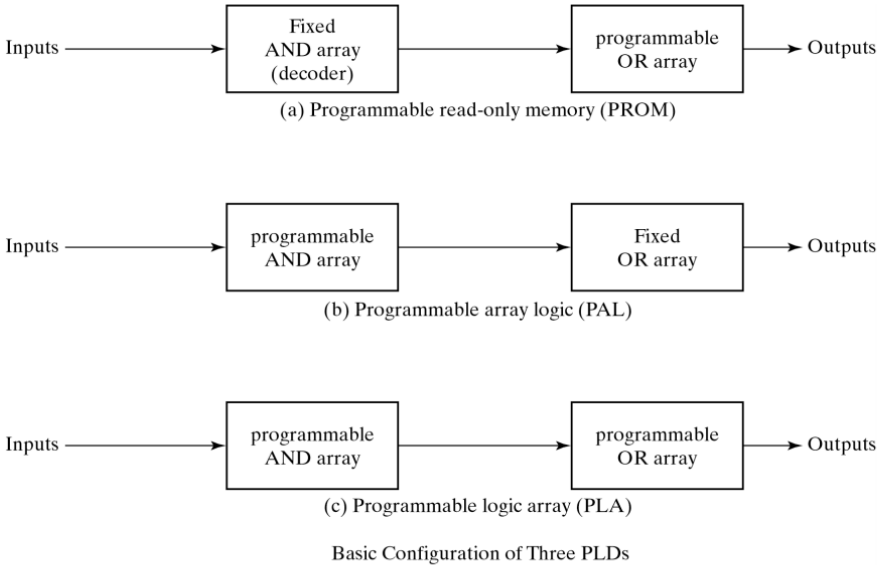
- Memory blocks can be (and often are) used to implement combinational logic functions:
- Examples:
  - LUTs in FPGAs
  - 1Mbit x 8 EPROM can implement 8 independent functions each of $\log_2(1M)=20$ inputs.
- The *decoder part* of a memory block can be considered a "minterm generator".
- The *cell array part* of a memory block can be considered an OR function over a subset of rows.



The combination gives us a way to implement logic functions directly in sum of products form.

Several variations on this theme exist in a set of devices called Programmable logic devices (PLDs)

PLD Summary



Basic Configuration of Three PLDs

Question 6:

Verification vs. Testing

- Verification is the task of ensuring that a design meets its specification
  - Looking for design mistakes
- Testing is performed to ensure that a particular instantiation of a design functions properly
  - Looking for implementation faults

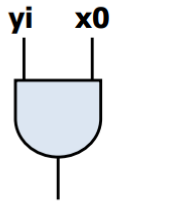
Types of Faults

- Transient faults
  - Faults that happen only once
    - and it's VERY unlikely to happen again
  - Causes:
    - Electromagnetic Interference
      - Neighbors mobile phone
      - Static electricity
    - Various particles hitting the silicon surface
      - Heavy ions such as iron, α-particles, neutrons.
    - Internal effects
      - Crosstalk, metastability, power supply disturbances
- Permanent faults
  - Faults that are always there
  - Causes:
    - Design defects
    - Manufacturing defects
    - Transistor aging
- Intermittent faults
  - Faults that come and go (probably periodically)
  - Causes: Variations
    - **Static**: transistors on a chip may not be exactly the same although they were supposed to be
    - **Dynamic**: temperature changes

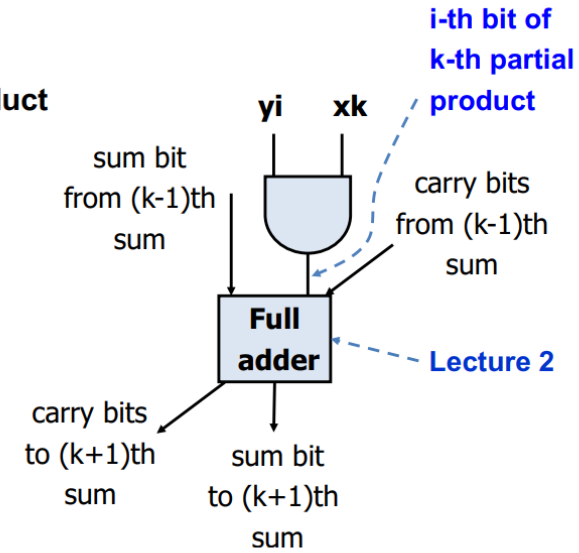
Question 7:

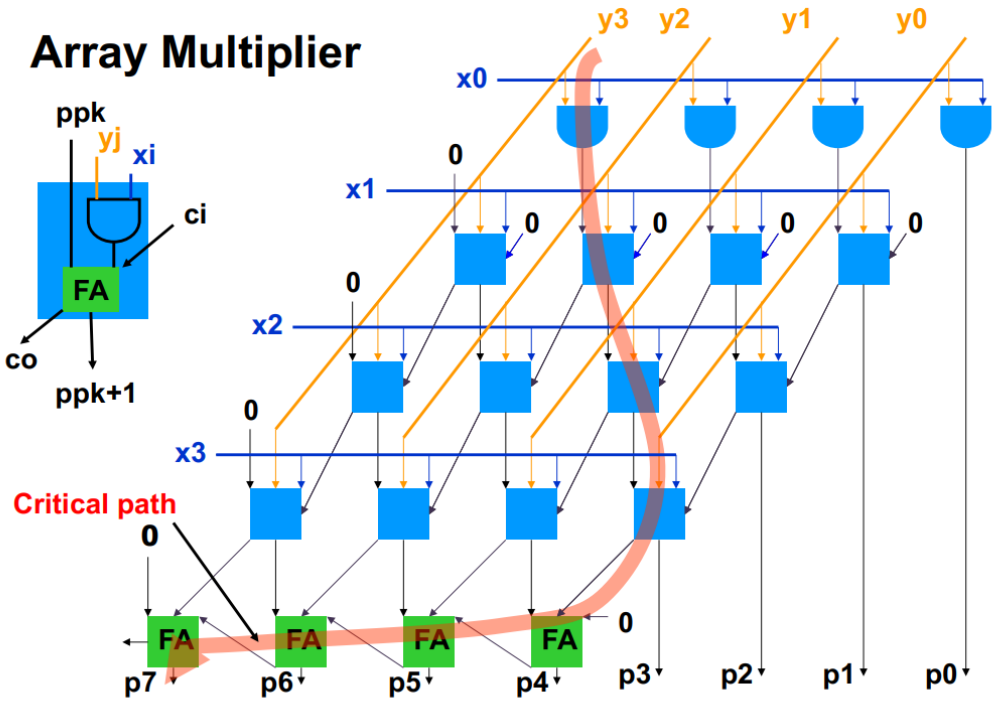
Basic Building Blocks

- Two-input AND
  - **for the partial product**
- Full-adder



0th partial product

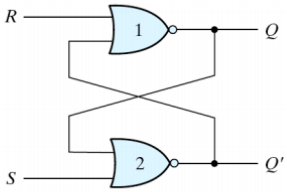




Question 8:

Analysis of an SR latch (1)

- We can analyze an SR latch using the previous technique



(a) Cross-coupled circuit

S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(After $SR = 10$)
0	1	0	1	
0	0	0	1	(After $SR = 01$)
1	1	0	0	

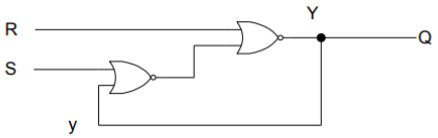
(b) Truth table

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

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

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

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



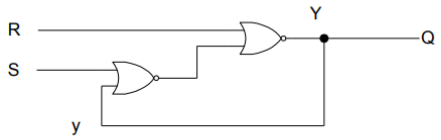
EDA322 Digital design, 2017-2018, Lecture 19
I. Sourdis, CSE, Chalmers
18

Analysis of an SR latch (2)

- Can derive the transition table and the flow table:

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

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



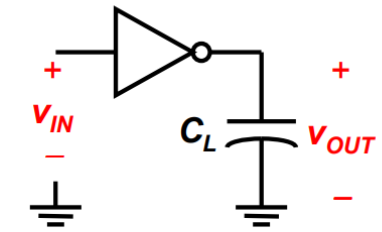
Analysis of an SR latch (3)

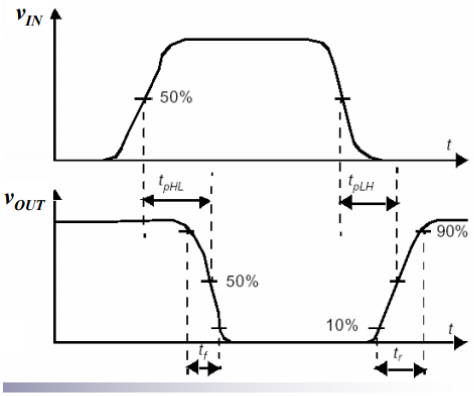
- Note: We can see the undesirable case when SR=11 and inputs change.
- Depending on the various delays and assuming SR=11 changes to SR=00...
  - If SR=11 -> SR=10 -> SR=00, we get stable state with output of 1.
  - If SR=11 -> SR=01 -> SR=00, we get stable state with output of 0.
- So the stable state is not predictable.
- Conclusion is that we need to be careful if we (possibly) need to transition from one state to another and we (somehow) pass through state 11.

Question 9:

Gate Delay

- The time needed for the output of a gate to change from the moment an input of the gate changes
- Depends on:
  - Transistor parameters
  - Fan-out: how many wires it will drive
  - Fan-in: number of inputs of a gate

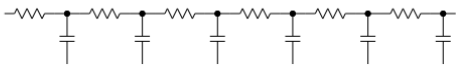




Wire Delay

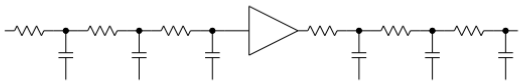
Even in cases where the transmission line effect is negligible:

- Wires posses distributed resistance **R** and capacitance **C**



- Time constant associated with distributed RC is proportional to the ***square* of the length O(L^2)**
- For short wires on ICs, resistance is insignificant (relative to effective R of transistors), but C is important.
- Typically around half of C of gate load is in the wires.

- For long wires on ICs:
  - busses, clock lines, global control signal (e.g. reset), etc.
  - distributed RC (and therefore long delay) significant
  - For long wires signals need to be "rebuffered" which contributes in the delay, too.



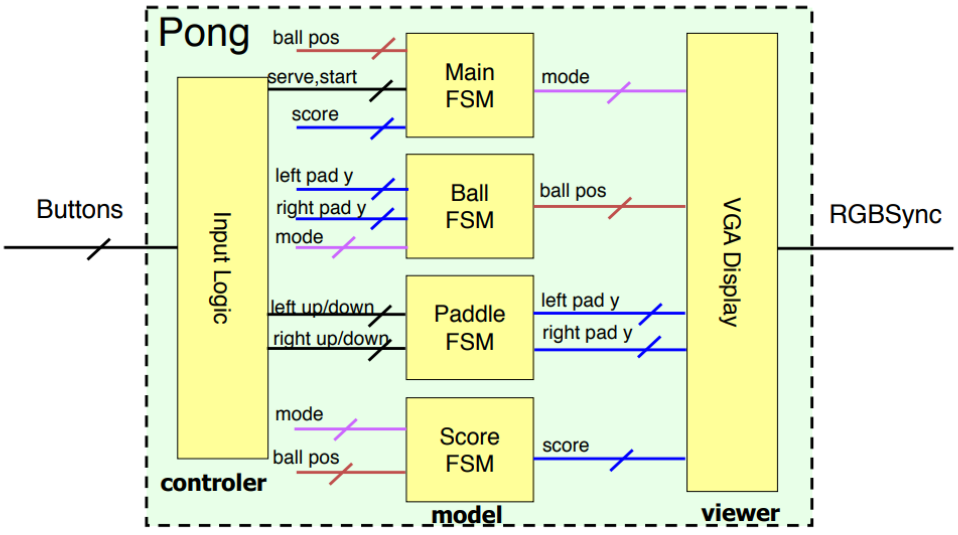
Wire delay is proportional to the ***square* of its length O(L^2)**

Question 10:

Divide and Conquer –common themes

- Task
  - Divide system into a network of tasks
  - One module per task
  - ***Model-view-controller***: tasks are:
    - The 'guts' (model)
    - Output modules that 'view' the model
    - Input modules that affect the model
- State
  - Divide system by state
  - Separate module for each set of state variables
- Interface
  - Module for each external interface

Pong Decomposition



Horizontally partitioned by task
Model partitions also vertically by state