EDA322/ DIT797: Digital Design
Exam - March 2019

Date: March 20, 2019

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

Examiner: Ioannis Sourdis

Department: Computer Science and Engineering

Inquiries: visiting the room at **15:30** and at
17:00
(contact through phone: phone extension 1744)

Results and grading review: room 4128 EDIT on **April 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.

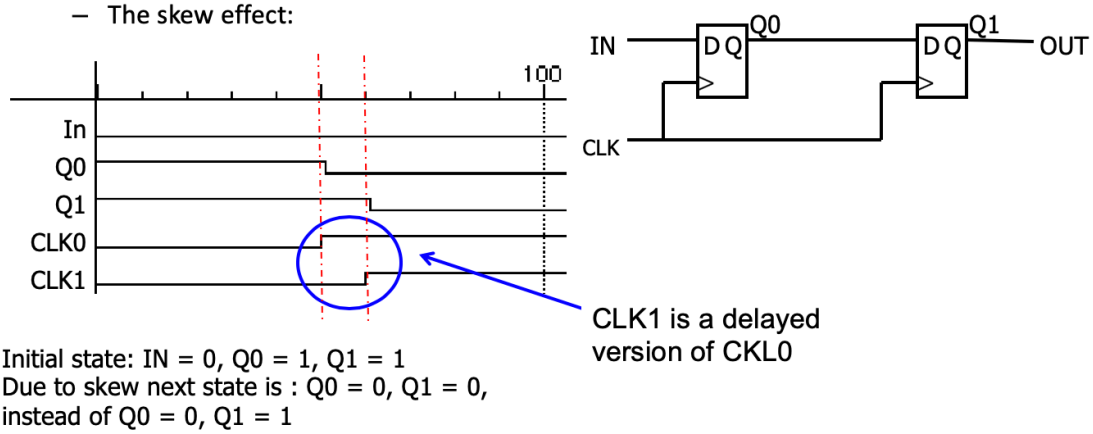
Your personal identity code is required on each submitted sheet!

Good luck!

Question 1 Sequential Circuits: (10 points)
a) Describe the problem of clock skew. What can we do to minimize its effects?
b) Describe the phenomenon of Metastability. When does it happen? What can we do to avoid it? What are the parameters that affect the probability of metastability in a flipflop?

Answer
a)

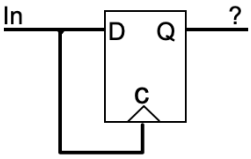
- The Problem
  - correct behavior requires that the next state of all memory elements are determined by all the memory elements thus at the same time
  - This is difficult in high performance systems since the time it takes for the clock to arrive, are of the same magnitude as the delay through the logic
  - The skew effect:



Initial state: IN = 0, Q0 = 1, Q1 = 1
Due to skew next state is : Q0 = 0, Q1 = 0, instead of Q0 = 0, Q1 = 1

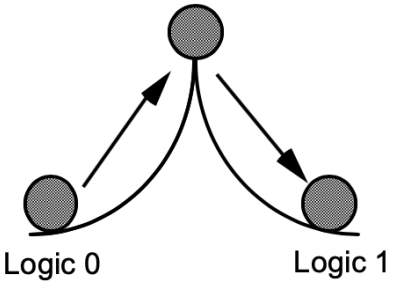
Clock distribution trees should be designed so all their leaves have the same delay (or multiples of a clock period), e.g. H-trees.

b)
Metastability may happens when an asynchronous input enters a flip-flop violating its setup and hold times. Adding one or better two cascaded flipflops to register an asynchronous input before it is used in a circuit, and choosing the parameters of these flip flops reduces the chances of metastability and the probability of failure. Besides the flipflop characteristics, the clock frequency and the rate at which the asynchronous input changes increase the probability of metastability in a flipflop.

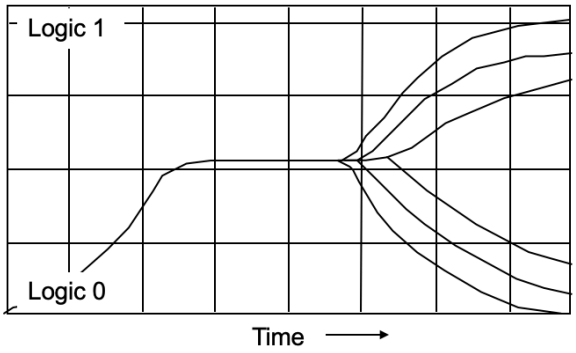


When FF input changes close to clock edge, the FF may enter the *metastable* state: neither a logic 0 nor a logic 1

It may stay in this state an indefinate amount of time, although this is not likely in real circuits



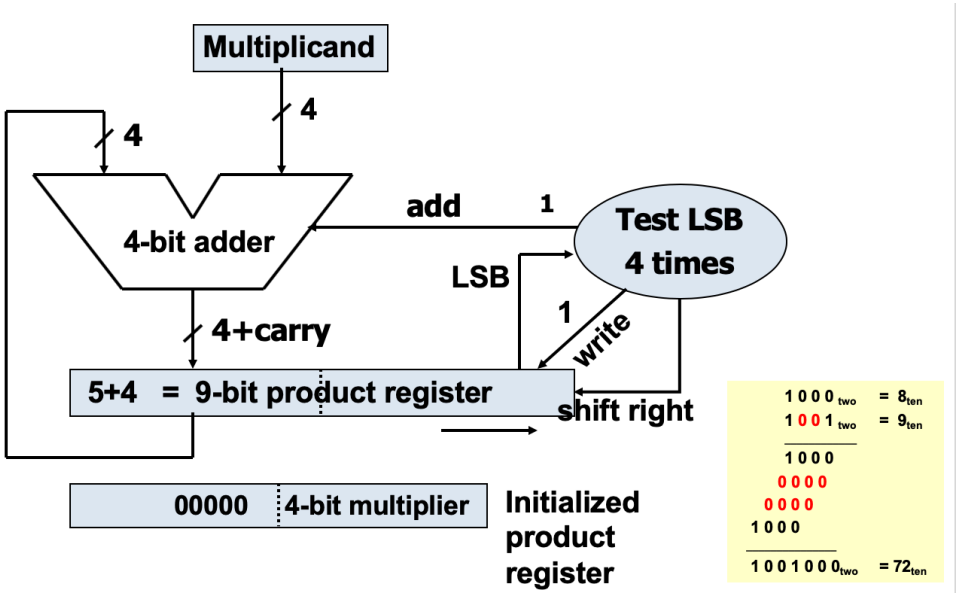
Small, but non-zero probability that the FF output will get stuck in an in-between state



Oscilloscope Traces Demonstrating Synchronizer Failure and Eventual Decay to Steady State

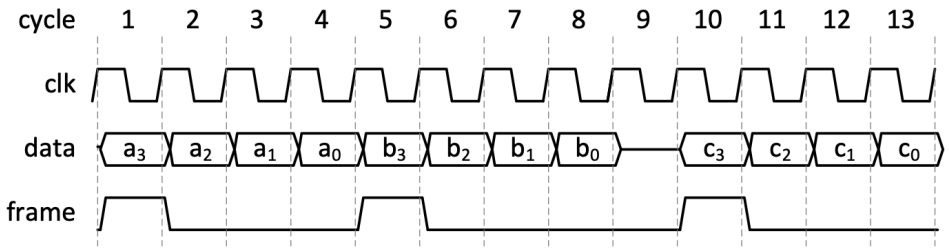
Question 2 Arithmetic: (10 points)
Draw the block diagram of a 4-bit serial multiplier. Explain how it will perform a multiplication (flow chart).

Answer



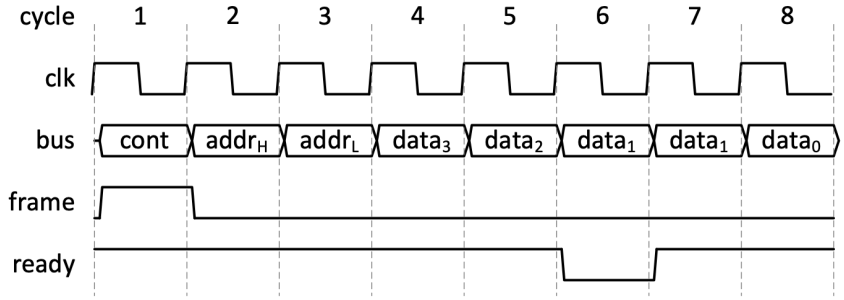
Question 3 Interfaces: (10 points)
Describe an Interface that uses serialization. What does it mean to have flow control at word granularity? Give an example of such as serialized interface.

Answer
When the data path is too narrow to send a word in a single cycle, a valid control signal indicates that a frame of multiple words will be sent (e.g. 4 words in the following example)

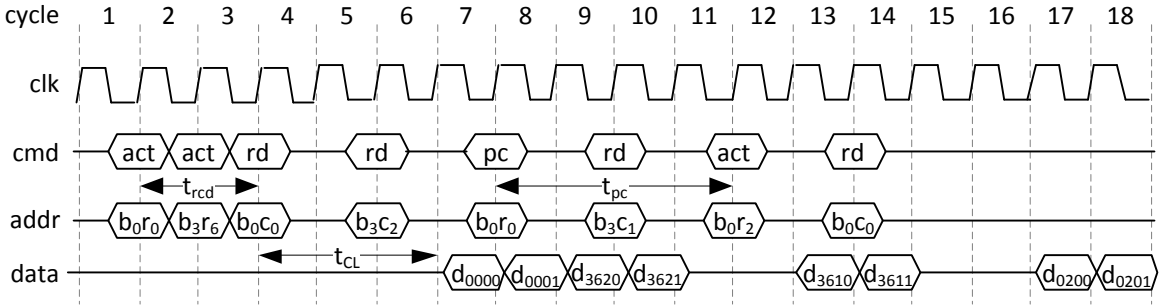


Frame signals start of new serial frame (in this case 4 words)
Flow control can be at frame granularity or word granularity

Flow control with word granularity means that the ready control signal send by the receiver refers to each word of the frame separately, e.g.:



An example of serialized interface is the interface of the DRAM where the data are send in multiple words through a bus.



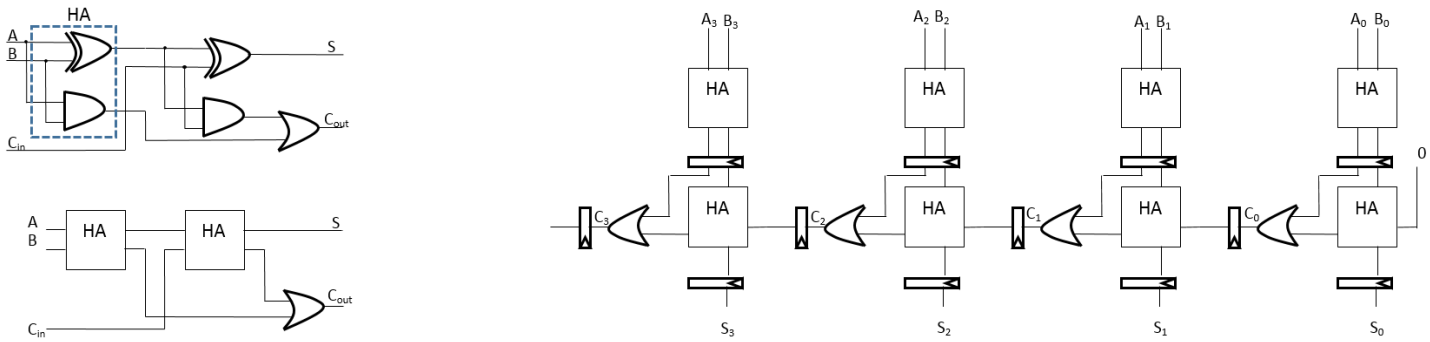
Question 4 Pipelining & Timing: (10 points)
Consider a 4-bit ripple carry adder and the following delays: 2-input XOR gate delay of 2 ns, 2-input OR or AND gate delay of 1ns, flip flop propagation delay 0.5 ns, flip flop setup delay 0.5 ns. In how many stages do you need to pipeline the above adder in order to improve its throughput more than 3 times? What is the latency of the pipelined and the unpipelined versions of the adder?

Answer

The initial ripple-carry adder has a delay of 10 ns and the throughput is 1/10 results/ns. In order to increase the throughput more than 3 times the stage latency needs to be at most 3.33ns: 3* (1/10) = 1/3.33

If each stage has up to 3.33 ns then 0.5+0.5 ns are gone due to propagation delay and setup time of the flipflop. The remaining 2.33 are less than the delay of a FA.

That means that we need to break each stage in to portions of logic that are smaller than a FA. One way to do it is to break the full adder in to two half adders as shown in the figure and then pipeline as follows:



The total number of stages are 5. Then, the delay of each stage is 0.5+2+0.5=3ns, the total latency of the pipelined adder is 5*3=15 ns, and its throughput is 1/3 results/ns.

Another simpler but less efficient way to pipeline the adder is to break each full adder in two stages: the first stage composed of only the first XOR gate and the second stage including the rest of the gates. That would require in total 8 stages of 3ns latency in each stage. Achieving the same throughput but with longer latency.

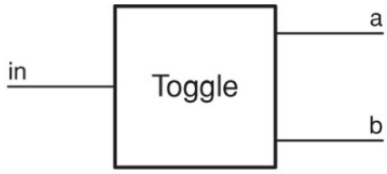
Question 5 FSM & Asynchronous: (10 points)
Toggle is a digital sequential circuit that receives one bit of input (A) and outputs two bits of output (Odd, Even). The first time the input is found to be 1 then the output Odd is also set to 1 and stays 1 until the input changes to 0, the output Even remains 0. The second time the input is set to 1 it is the turn of the Even output to be set to 1 and remain 1 until the input becomes 0, the Odd input remains 0. In general, the output odd follows the odd pulses of the input, and the output even follows the even pulses of the output.

a) Design the Toggle module with a synchronous sequential circuit considering D-flipflops. Draw the timing diagram of the Toggle to show how it will function. Then draw the state diagram, state table, derive the Boolean expressions and draw the gatelevel of the Toggle.
b) Now design the Toggle module with an asynchronous sequential circuit. Draw its timing diagram to show how it will function and compare it with the synchronous version. Then, draw the, state table, derive the Boolean expressions, protect against hazards, and draw the gatelevel of the asynchronous Toggle.

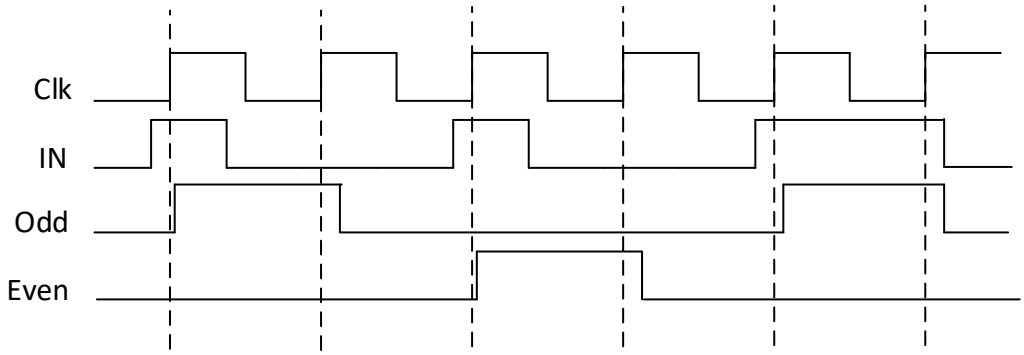
Answer

a)

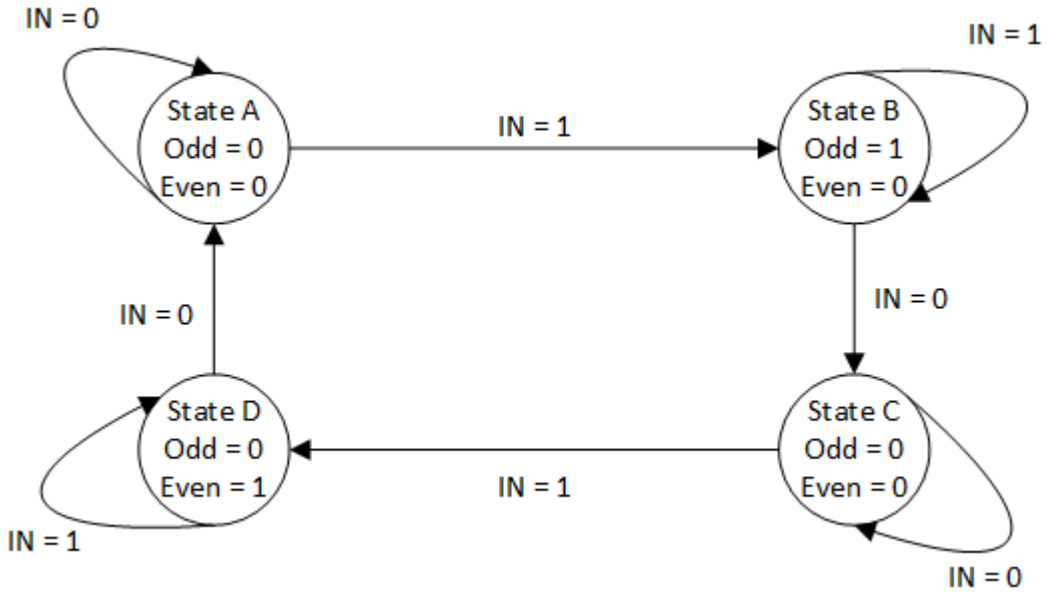
Toggle module:



Timing diagram:



State diagram considering a Moore FSM:



State table:

State	Input / Next State		Output	
	IN = 0	IN = 1	odd	even
A	A	B	0	0

B	C	B	1	0
C	C	D	0	0
D	A	D	0	1

State table with binary codes assigned to each state:
A:00, B:10, C:11, D:10

State S(1)S(0)	Input / Next State		Output	
	IN = 0	IN = 1	odd	even
00	00	01	0	0
01	11	01	1	0
11	11	10	0	0
10	00	10	0	1

Karnaugh maps for next state:

N(0):

S(1)S(0)	00	01	11	10
IN = 0	0	1	1	0
IN = 1	1	1	0	0

N(1):

S(1)S(0)	00	01	11	10
IN = 0	0	1	1	0
IN = 1	0	0	1	1

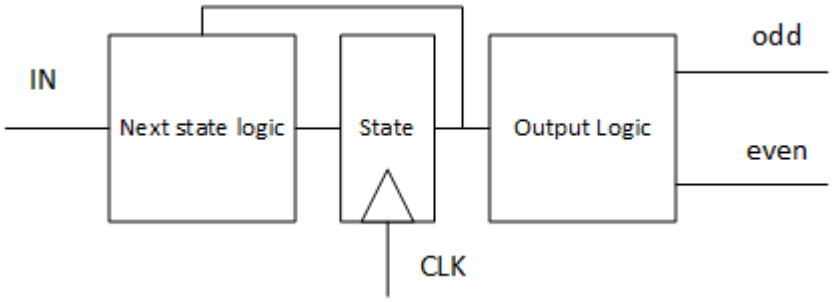
Boolean equations from Karnaugh maps:

N(0) = IN' * S(0) + IN * S(1)'
N(1) = IN' * S(0) + IN * S(1)

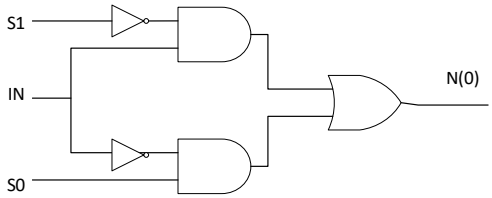
Outputs only depend on state (moore FSM):

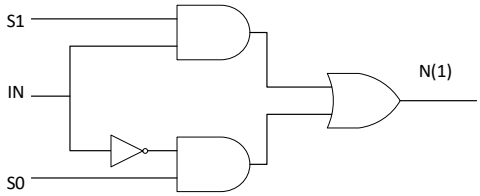
Odd = S(0)*S(1)'
Even = S(1)*S(0)'

Gate Level

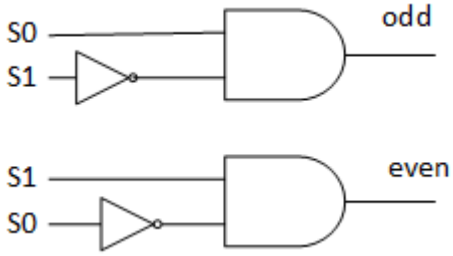


Next state logic:



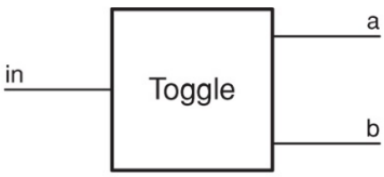


Output Logic:

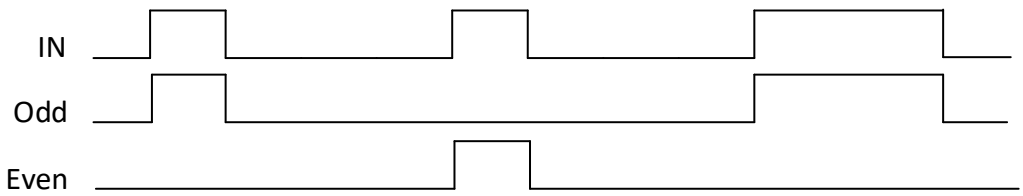


b) As is in lecture for asynchronous, slides 40-43 + a drawing of the gatelevel design.

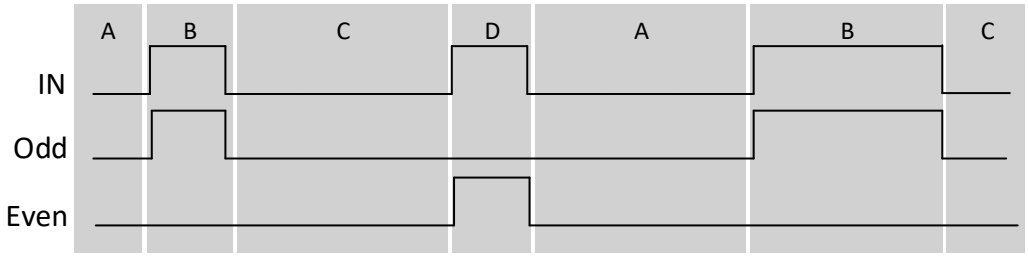
Toggle module:



Timing diagram:



State diagram:



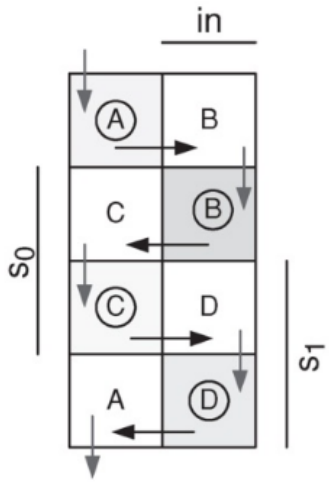
State table:

State	Input / Next State		Output	
	IN = 0	IN = 1	odd	even
A	A	B	0	0
B	B	C	1	0
C	C	D	0	0
D	D	A	0	1

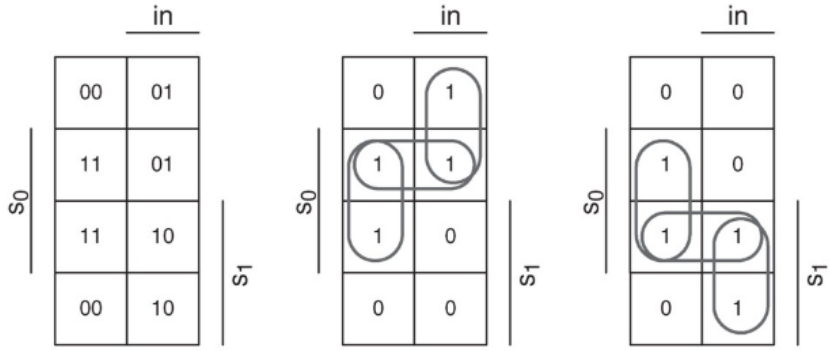
State table with binary codes assigned to each state:

State S(1)S(0)	Input / Next State		Output	
	IN = 0	IN = 1	odd	even
00	00	B	0	0
01	01	C	1	0
11	10	D	0	0
10	11	A	0	1

Flow table as Karnaugh map showing the state transitions (Trajectory map):



Separate maps for each next state variable:



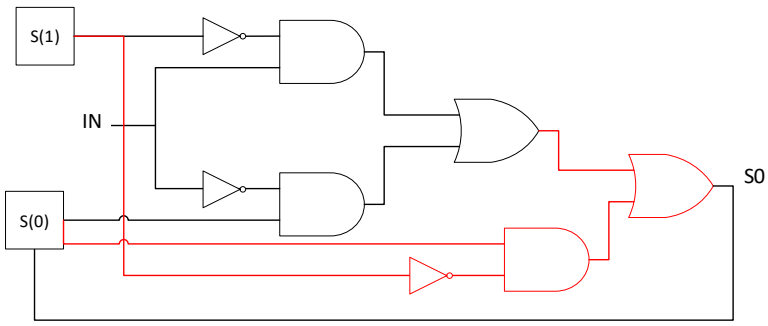
Boolean equations from Karnaugh maps:

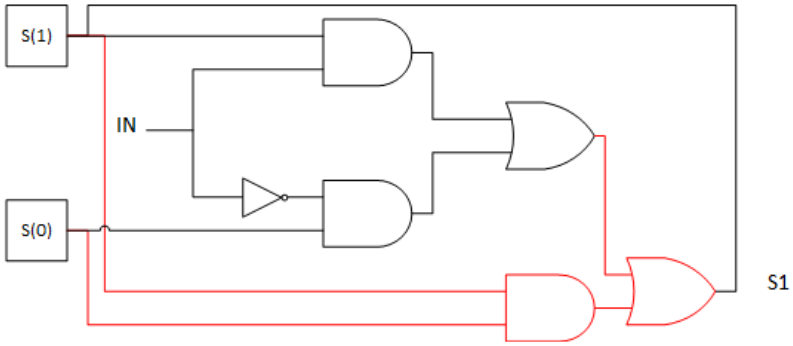
Next state:
s0 = s1'*in + s0*in' + S(0)*S(1)'
s1 = s1*in + s0*in' + S(0)*S(1)'

Last Implicants to avoid hazards

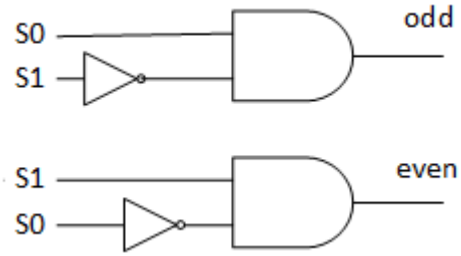
Output equations:
a = s1'*s0
b=s1*s0
Gate level:

State:





Output:

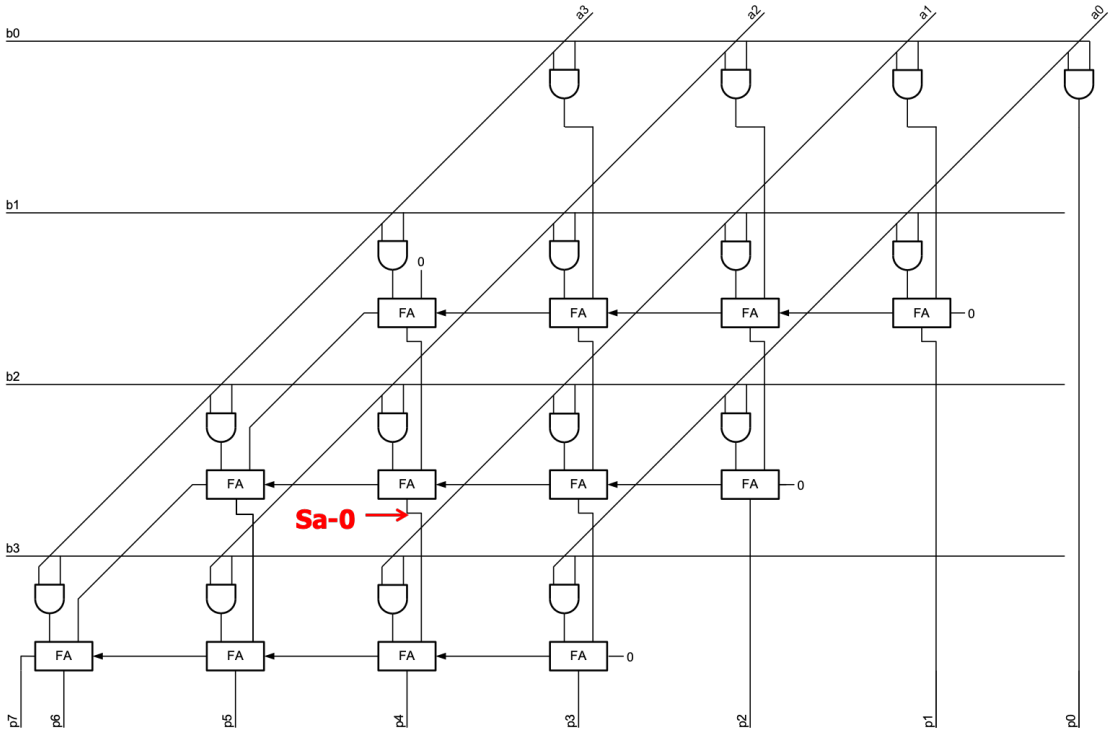


For b) see also lecture for asynchronous, slides 40-43

Question 6 Arithmetic & Testing: (10 points)
Draw the block diagram of a 4-bit array multiplier. Consider that the two binary numbers multiplied are A[0-3] and B[0-3]. Show how to create a test for a stuck-at-0 at the "summary" output of the full adder located in the cross-section of A[2]-B[2].

Note: a full adder can be used as a black box without showing its internals.

Answer



B0=0, B1=0, B2=1 and A0=0, A1=0, A2=1 are inputs that will activate the fault (the sum of the FA will be "1" which is the opposite value from the stuck-at-0)

Then, with B3=0 the fault will be propagated to the output P4.

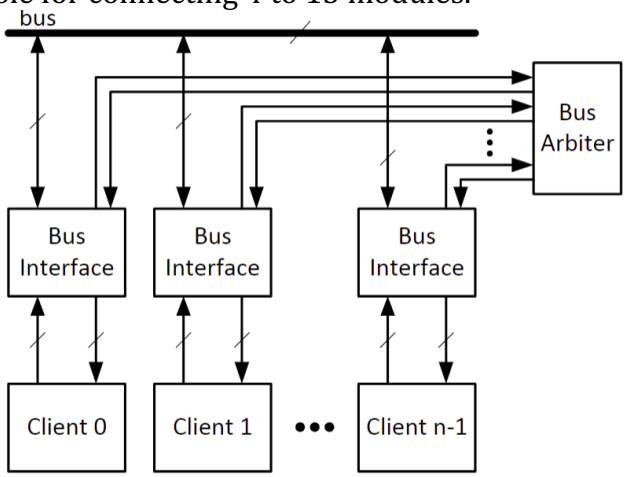
Question 7 Interconnects: (10 points)
What are the two performance criteria based on which one would choose a type of Interconnect on a chip under design? Describe 2 types of interconnects and explain how they would perform based on these criteria. How would they compare in terms of area (chip resources)?

Answer

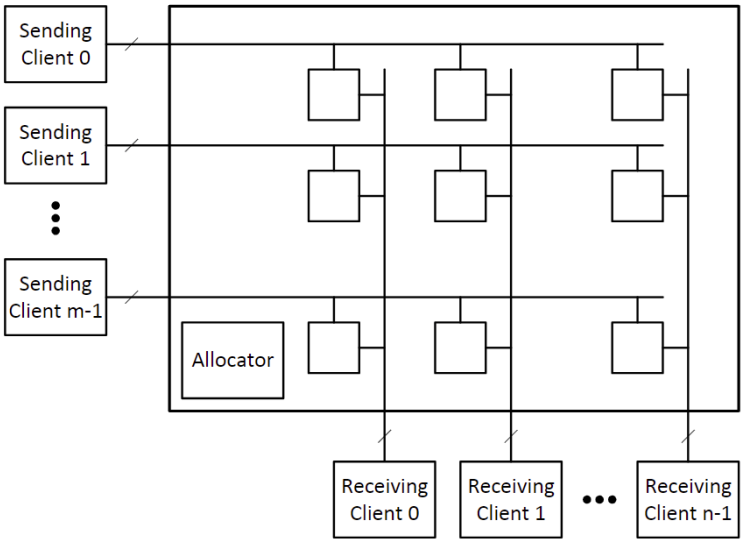
Performance Criteria:
Packet latency: the delay of a single message to go from a sender to a receiver
Throughput: the average amount of messages delivered per unit of time.

[one can select any of the following two]
Shared Bus:
A shared bus consists of a single set of wires which connect all the communicating modules on a chip. Only one module can send data over the bus at a time (although data can be received and used by one or more different modules. A bus arbiter is used to grant permission to requesting modules to access the bus for a fixed duration of time.
A shared bus is not scalable to connecting too many communicating modules (clients). This is because increasing the number of modules communicating through a bus increases the capacitive load on the bus which slows the clock

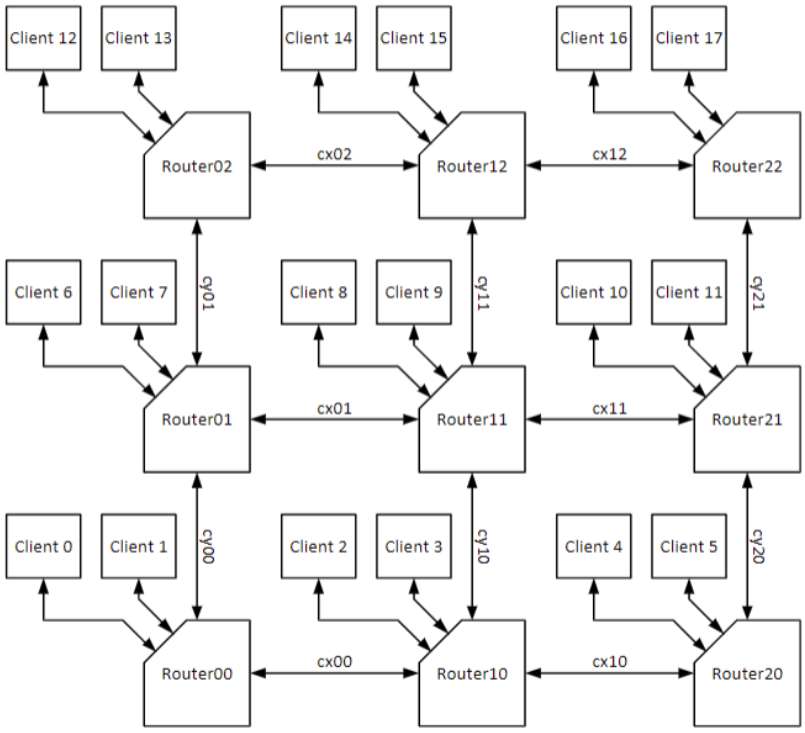
frequency of the bus. This increases the message latencies over the bus and also reduces the throughput.
It is usually suitable for connecting 4 to 15 modules.



Crossbar:
A crossbar is a switching element composed of either multiplexers or tristate buffers. It uses a scheduler (allocator) to select a set of masters and connect them to different slaves for a fixed duration of time.
A crossbar allows multiple modules to communicate concurrently while a shared bus only allows a single connection at a time. Compared to a bus, this reduces contention among the modules trying to access shared resources, and consequently reduce messages latencies and increase throughput over a crossbar. As the number of modules connected using a crossbar increase, the operating frequency of the crossbar switch decreases. This is because of the need for larger multiplexing logic and longer wires to connect the crossbar ports with the modules. Crossbars have high area costs (increasing by $O(n^2)$ to the number of connecting modules) and are usually suitable for connecting 16 to 64 communicating modules.



Mesh Interconnect:
A 2D mesh interconnect is composed of small routers connected with their neighbors in X and Y directions using link wires. Packets enter the mesh network at the source node, propagate through multiple routers on their way to the destination node and are ejected out of the network at the destination node.
Mesh interconnects also support concurrent communication among nodes. Mesh networks can offer higher through and lower message latencies compared to busses but not necessarily compared to crossbars.
Mesh interconnects are more scalable compared to crossbars and buses and are suitable for connecting over 64 communicating modules.

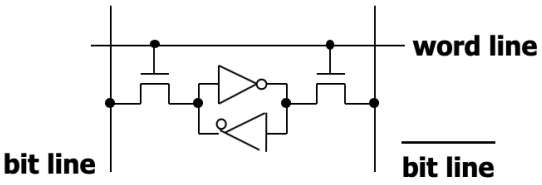


Question 8 Memory: (10 points)

- Draw an SRAM and a DRAM memory cell.
- Explain their advantages and disadvantages.
- Describe the process of reading and writing to these cells.

Answer

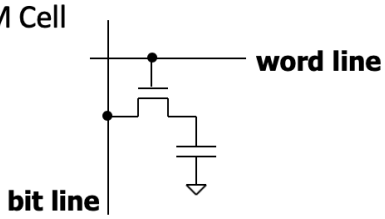
- SRAM Cell



- Larger cell ⇒ lower density, higher cost/bit
- No refresh required
- Simple read ⇒ faster access
- Standard IC process ⇒ natural for integration with logic

- **Read:** Pre-charge bit-lines, Raise wordline, cell pulls one bit line low, sense the difference
- **Write:** Drive data onto bit-lines (one 0, one 1), Raise wordline

- DRAM Cell



- Smaller cell ⇒ higher density, lower cost/bit
- Needs periodic refresh, and refresh after read
- Complex read ⇒ longer access time
- Special IC process ⇒ difficult to integrate with logic circuits

- **Read:** Pre-charge bit-lines with Vdd/2, Raise wordline, bit-line gets the read value, value sensed, and written back to the cell
- **Write:** put the value to write in the bit-line, Raise wordline
- **Refresh:** a dummy read to every cell.

Question 9 Number representation: (10 points)
Suppose you need to represent time from 1 nanosecond (ns) to 1000 seconds (sec) with an accuracy of 1%

a) What type number representation will you choose and why?
b) Find the exact number representation, how many bits are required?

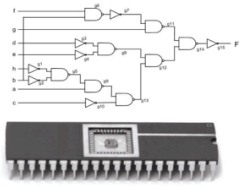
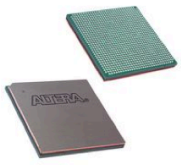
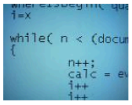
Answer

a) Floating point is a better choice because the required accuracy is a percentage and not a fixed absolute value.
b) A fixed-point representation would require 46 bits (10.36). That is 10 integer bits to count from 0 to 1000 seconds and 36 bits of fraction for having accuracy of 10ps (1% of a ns), which is 20 ps of resolution.

   For a floating-point representation, the mantissa needs to be only 6 bits to offer accuracy of 1% 1/100. So, the resolution needs be double that: 1/50; $1/64 = 1/2^6$ should then be enough (6 bits mantissa). The dynamic range is 10^{12} (1000 sec = 10^{12} ns) which is smaller than 2^{40}. So, 6 bits exponent is enough to represent 2^{64}. We finally set the bias to 54 so the maximum exponent to be 2^{10} = 1024 since the largest number is 1000 seconds.

Question 10 ASIC and Reconfigurable Technologies: (10 points)
Explain what are the advantages and disadvantages of (i) ASICs and (ii)software running on general purpose computers. Where does the Reconfigurable hardware stand between the two?

Answer

Hardware (Application Specific Integrated Circuits)	Reconfigurable computing	Software-programmed processors
		 
Advantages: • very high performance and efficient **Disadvantages:** • not flexible (can't be altered after fabrication) • High NRE Cost	**Advantages:** •much higher performance than software / lower performance than ASIC • higher level of flexibility than hardware / more difficult to program than SW **•fills the gap between hardware and software**	**Advantages:** • software is very flexible to change **Disadvantages:** • performance can suffer if clock is not fast • fixed instruction set by hardware

END of EXAM