Timeline and expectation:

Jan 30, confirmation of project and group (group of 4)

Feb 15, submit specification sheet and block diagram and outline all state machines.

March 31 submit final code

April 1-10 demos.

Meet the lab instructor in room 208 kresit bldg once every 2 weeks at 5 pm on Tuesday i.e. after your class to discuss the project for 10 min or so.

1. Design an aircraft control system.

Ramesh is a 10 pointer who has been given special access to the best hostel on campus (14) which is 1.5 km away. He is ill, and is unable to deliver the assignment to Professor A. Ramesh does not want to send his assignment through someone and Prof. A is particular that the assignment must be delivered in hard-copy. So Ramesh hires you to give him an aeromodel that would take his assignment from his Hostel terrace and give it to Prof. A just outside his office.

The inputs to the aeromodel are: elevator up, down, aileron up, down, throttle open or close in 10 increments/decrements, rudder left or right.
The aeromodel consists of 2 kinds of sensors: a far sensor and a 360-degree sensor.
The far sensor computes the distance to the nearest obstacle in the flight-path while the 360 degree sensor computes the distance all across the plane in every direction and stores the closest 3 obstacles and their orientations wrt the flightpath direction.

Whenever there is an obstacle, the plane attempts to make an upward role or find another direction. It takes a finite amount of time to change the direction, and 1/3 this amount of time to initiate a roll. The plane must land outside the profs' office. The landing speed must be less than 3 increments of the throttle. You can assume the plane has thrust vectoring using 2-engines. http://en.wikipedia.org/wiki/Thrust_vectoring Thrust vectoring can make the plane undergo any motion you desire.

Note that the path from Ramesh's dorm room to the office is fed into a memory chip. The plane also has a 1-meter resolution GPS built in it.
What you have to do:

Write all the state machines that will suffice Ramesh to make his aeromodel deliver the assignment to Prof A.
Write test benches with real data taken from google maps so that you can measure how much time it takes.
You can be more creative and assume Ramesh to stay somewhere else - his home town or anywhere. For this his plane has to take refuelling halts.

Assignment would be graded based on creativity.

2. Design and implement a memory processor interaction in RTL.

Assume that you want to write two pieces of code at the memory and at the processor that would be hardcoded as IPs in both the chips.

The protocol is as follows: The processor gets 4 lanes of data from different internal units with the lane-widths being 8, 4, 4, and 12 bits. The lanes are at different speeds with the 12 bit lane at 1.5x the speed of all other lanes.

The wires between the processor and memory are the same for both sending from the processor to the memory as well as receiving from the memory by the processor. The memory is arranged into 8 banks or 40 word depth each and each word is 128 bit long.
The memory has two pointers one to select the bank number and the other to select the word number (after the bank is selected). The memory works at 250 MHz.The processor works at 1GHz. To make up for the mismatch in communication, the memory and processor use a FIFO at both ends. The processor stores data in the FIFO prior to sending it to the

memory. Similarly, whenever the memory has to send a word or receives a word, it stores the same in the FIFO.

You may use coregen to design the FIFO.
You are required to design and implement the protocol in an efficient way.
You must create a work conserving system with suitable assumptions (will be explained in clasS).
You must choose the number of IO lines between the memory and the processor. A goal of this design is to minmize the IO lines.

Note: a word is up to 128 bits long, but could also be shorter. It may however not be less than 20 bits long.

Show through a test bench how you would measure system performance.
If throughput is defined as the amount of time the system is busy as compared to the amount of time it is free, then compute the throughput for your protocol.
A key feature of the protocol is the fact that the same wires are used for read and write to the memory.
You may use a wr_en signal that tells the memory that the CPU is going to write into it.
You may use a rd_en signal that tells the memory that the CPU is going to read from it.
You could use a RQ_t_send signal that fetches data from the memory
You could use a comp_send signal that tells the memory that the CPU is now finished with sending data.
These signals are called control signals and these are beyond the data lines and wont be counted towards your optimal number of lines computation exercise.

3. Design a multifunction elevator

The KRESIT elevator is broken. You are to design a controller in RTL for the elevator.
To begin with there are 5 levels, and the elevator has to follow the following constraints:
If a caller outside presses for the elevator to go up or down, the elevator shall respond with priority with its own direction of movement. Implying that the elevator shall continue its motion and record the callers requests giving it priority over another request if that other request is in the opposite direction of motion.The amount of time the doors require to close is dependent on the weight of the elevator. If there are more people in it, the door closes quickly, else it takes a slightly longer time to close.

The elevator has a fire alarm button that when pressed makes the elevator go to the nearest floor and open its doors. For purpose of audit the elevator stores all its movements in a memory. Create this memory system.

Design your system with the following constraints: for every floor traversed up, the energy required is 3 units; for every floor traversed down, the energy required is 2 units. The doors should wait for 5 seconds per 100 kg to close and the max weight possible is 400 kgs.
Compute the total amount of energy required by your system for the following traffic scenario

Load	at	Up	Down	Weight
2	2	1	0	100
3	4	0	3	210
4	2	2		320
3	3	0	2	175
9	0	5	0	650

4	4	3	0	250
12	0	5		960
3	4	3		180
2	1	3		150
2	4	1		180
2	3	1		120
7	4	3		500

4. Design a traffic management system at the three traffic lights that impact IITB, i.e. market gate, main gate and the pizza hut junction.

Assume that the volume of traffic that enters market gate junction is 100 cars per minute from eastern expressway, and the volume of traffic that enters the pizza hut junction from JVLR is 120 cars per minute while the amount of traffic that enters the pizza hut junction is 15 cars per mnute from Hiranandani.

Assume that the traffic from IIT into the main gate junction is 5 cars per minute.

Assume that the traffic that enters the main gate junction from the south, i.e. from Powai village is 8 cars per minute.

The number of lanes on JVLR are 4 for West bound traffic, 5 for east bound traffic and service lanes of two lanes. The market gate and maingate are all single laned. The traffic from powai village is single laned and you can assume that the two lanes coming in from powai village are both merged into a single lane. The number of lanes going into HN is 2, while the number of lanes coming from HN are 3. We do not care how much traffic goes into HN, but are aware that it is less than the traffic that comes out of HN at the pizza hut junction.

Now design the three traffic lights through state machines.

To aid your design, there are wire coils on the main JVLR for east and west bound traffic and we know how much traffic is backed up. Two inputs are available - average speed of traffic at the market gate and the pizza hut signals and number of car rows of backed up traffic.

Create linear equations for measuring the flow of traffic.

Reduce these equations and map them to state machines at the three traffic lights.

Factor in pedestrian movement.

Your code must never lead to an infinite queue.

Compute max waiting time at a traffic light and mean time through the three lights. Assume distances as 800 m between market and maingate, 800 m between main gate and pizzahut.

Code in VHDL and create a test bench.

AP problem: now if the traffic rate continues to change, how will your solution change?

Define peak load as the number of vehicles handled in the east-west corridors and compute peak load for the system you designed.

AP: Now, if you were to redesign the system what new rules would you enforce so that the peak load is enhanced. What is this new peak load?

5. SDN controller

You have to design a software defined network controller.

After every 20 clocks you get an NxN adjacency matrix whose element A(i,j) if = 1, tells you whether an edge exists or otherwise. After every 200 clocks you get a traffic matrix of the order NxNxkxm where N is the number of nodes in the network, k is the number of service levels and m is the number of service instances at that service level for the particular source-destination pair under consideration.

When a traffic request arrives - you have to compute the shortest path in an application and pass on a string that emulates the nodes from the source to the destination.

You have to resolve deadlocks in a timely manner.

When you send the routing string the switch has only capacity for 10 such strings. Hence if a 11th request comes, then you have to delete the 1st request and replace it.

Keep all the routes that you populate with the switch in the off FPGA memory only.
Design and implement a memory controller that is work-conserving.
The controller should be able to also create routes for protection that is edge disjoint from the work path. When a work path fails the destination node intimates the controller. The controller then populates the tables with the protection path.
The controller also informs switches of priority of handling services.
The controller also measures statistics such as average latency for each path, jitter and packet drops. It does so by sending a packet to the network with some timestamps (assume all switches can adhere to timestamps and have a central sync clock).

6. SDN switch design

In this project you are to design a switch that can interface with a controller.
You have to design an NxN switch. The switch should support a crossbar structure
The switch works as follows:
Packets enter the switch as 8 bits at a time for each of the N lines at 125MHz. A packet that enters the switch comprises of two parts – a data part of 6400 bits and a control part of 48 bytes. The first 6 bytes are proprietary and are generated as a fixed sequence of 10110101 repeated 6x.
The next 4 bytes contain the destination address and next 4 bytes contain the source address, and then the remaining bytes in the header are used for storing tags.
When a packet first comes into the network, it is first checked against a table which should have the destination address value. If the destination address value is not in this table, then there is a finding engine which talks to the controller and fetches the value in 20000 clock cycles. Till then you must buffer the packet. Once a match is obtained between the table and the packet's destination the match yields a TAG. This tag is 32 bits long and is inserted in the packet.
The packet is then sent to the next logic block that examines the TAG value from all the packets arriving in this logic block (called contention and scheduling block). The contention and scheduling block (CSB) is connected to a NxN switch fabric that is built using either VOQ or VIQ principles. VOQ implies virtual output queuing, while VIQ implies virtual input queuing. The switch fabric is to be built using multiplexers.
The CSB has access to buffers such that these can store the packet in case the input or output or both ports to the switch fabric are busy.
There are two ways to implement the CSB and switch fabric. In one way, you can store packets in buffers while the ports of the switch fabric are busy. In another way, you can store all packets in a buffer and only switch the header and once the header arrives at the destination port, then this destination port will fetch the packet from the memory.
The goal of choosing either implementation should be that the design is (a) as work conserving as possible and (b) the design with lowest possible resources.
Will your design change if you did not have to worry about buffers? Justify your answer.
Design a testbench to test the system.

7. Agro-dispenser and indicator

Rahim works as a farmer and has a huge expanse of land. He wants to design a do-it-all agro machine. The machine has the following functions:

- Automatic watering system
- Seed dispensor
- Grass cutter
- Soil tiller
- Field health monitoring

The agro dispenser has the following inputs:

- It can check the soil humidity and comes into it as a 8 bit vector. Depending on the level of humidity and the type of crop you have to keep the soil humid by watering the field.

- Whenever the crop is grown to a level you have to cut it. Crop height is measured as a 3 bit vector. Only when the crop height grows beyond 111 do you cut it. Unless the crop has grown and the field health monitoring system says that the crop is dead, in which case you cannot cut it but have to report it before cutting it.
- When the land is barren for 30 days, you have to water it and then sow a bunch of seeds. There are 8 types of seeds and you must make sure that the right type of seed is sown i.e. there is local homogeneity in the crop (though you may divide the field into 8 or more divisions).
- When you cut the crop you till the soil. You do this for 3 alternate days. You till an area.
- The dispenser moves at 1m/min speed and motion is only along the X or the Y axis.
- The cutter cuts one crop per second.

The following facts are established:

Crop type	Rate of growth	Variance	Min # per sq meter	Dur
W	4	3	80	90
R	1	3	80	120
S	6	2	120	180
B	3	2	40	40
A	1	4	20	80
M	1	2	1	720
G	4	2	4	360
O	4	1	8	1000

For each action 2 watts of power is consumed.
Design an automated algorithm that takes care of Rahim's field.
Compute the power required per year. (dur col = days in the table above)

8. Missile evasion system

In this project we want to design a missile evasion system that sits in an aircraft that flies through enemy territory.
A missile on the ground locks on to your aircraft and is fired. The missile has a peak speed of 2000 km/h, while your aircraft can go only up to 1200 km/h. the missile cannot slow down below 1600km/h and the rate of making turns for the missile is up to 15 degrees in 3D space at 2000km/h and 30 degrees at its slowest speed, following a linear function.
The aircraft can make turns at up to 60 degrees at slow speed and up to 22 degrees at peak speed. The initial distance when the missile is detected is 4 km.
Design a state machine based evasion system that facilitates the aircraft to successful evade any such missile.
The aircraft has 7 controls that can be controlled: rudder right/left, elevator up/down, throttle high/low and breaks ON (release is default).
Design a testbench to test your system.
Build enough cases so that you can avoid the missile.
The missile is avoided when it misses your aircraft thrice in any plane of motion. Alternatively if the missile explodes in the distance 300-500 meters away from your aircraft you have still

avoided the missile. The missile is designed to explode 100-400 meters away from your aircraft it has a max range of 12 km.
Assume all functions are linear.

To prove that you are able to avoid the missile, plot the missile and aircraft's 3D coordinates (x,y,z).

Suggest a method so that the experiment is repeated and with a high possibility we are able to avoid the missile. Suggest how you can compute the possibility as a probabilistic measure.

9. Desir chef

You are to design and implement a food processor. The food processor takes in the following ingredients and despatches the corresponding foods and uses the corresponding apparatus:

Food	Ingredients	Apparatus	Functions	Times
Bhaji	Potatoes, onions, chopped tomatoes, chopped garlic, chopped cilantros, cauliflowers, peas, masala, butter	Masher, cooker, grater, smasher, tava	Takes in potatoes and boils these. In parallel, takes in chopped onions, garlic and fries these on a tava for 15 mins. An ML IP algorithm tells you once these are golden brown. Subsequently add diced tomatoes and fry for another 8 mins. Add the masala by one portion @ of the spice level. Add boiled peas and potatoes and crushed+boiled cauliflower. Stir, grate the mix and smash the bhaji. Subsequently add butter.	
Paneer Makhani	paneer, chopped onion, chopped tomatoes, chopped garlic, masala, cream, dahi, peas, oil	fry, grater, mixer, slow cooking	Fry chopped onion, rye, jeera proportional to the chopped onion. Add tomatoes once the ML algo tells you that the onions are golden brown. in case of jain prep, add cashew gravy. Add chopped garlic. Fry for another 9 mins. Grind the fried mix after cooling it to 60 deg C. Then add the masala based on the taste level. Slow cook with chopped and grated paneer for 20 mins.	
Alu sabzi	Potatoes, mustard, rye, oil	chopper, shallow fryer	WILL FILL LATER	
Bhindi masala	Bhindi, oil, onions, mustard, rye, (whats rye in english?),	Grater, shallow fryer, slow cooking	WILL FILL LATER	

Batata wada	Potato, oil, besan, condiments	cook, boiler, mixer, deep fryer.	WILL FILL LATER	

Design a single state machine for all of the above with IO diagram as below. Each food item comes in three flavors - spicy, medium and (non spicy). In addition a food would also be categorized as jain/non-jain. For jain selections you would not add onions and garlic.

One of the design goals must be to reduce the wastage of food.

All food dishes are measured by a standard plate size and the input ingredients are taken as a measure of this standard plate size (computed ahead in time as per the function column in the table above.

For each apparatus, 2 units of electricity are required for create any dish. Hence bhaji with 5 apparatus requires 10 units of power. You have to calculate and display the power usage in a day, resetting it every month.

Create a testbench to check your design.

Implement this on an FPGA based board and show the output through streaming LEDs.

SAT solving:
In this problem, we want to design a hardware SAT solver for Boolean formulas. A SAT solver takes as input a Boolean formula in a specified format (explained below), and reports one of two things: (i) UNSAT, if there is no assignment of truth values to the variables in the formula that renders the formula True, or (ii) SAT if there is an assignment of truth values to the variables that renders the formula True. In the latter case, the solver also produces as output an (any) assignment of truth values to the variables that makes the formula evaluate to True.

In this problem, we are going to consider Boolean formulas over 64 or fewer variables. Every input formula is assumed to be in conjunctive normal form (CNF), which is defined as follows:

a. A literal is a variable or its negation
b. A clause is an OR of literals
c. A CNF formula is an AND of clauses

For example, if a1, a2, a3 are Boolean variables, then
f1 = (a1 + a2) . (a1 + a2 + a3) . (a3 + a2) is a Boolean formula in CNF with 3 clauses, while
f2 = a1.a2.a3 + a2.a3is certainly a Boolean formula, but not in CNF.

The digital circuit you must design should have the following binary inputs:

1. Load: When this input turns high from low, a new CNF formula is to be read in. The reading in of the formula continues until Load remains high.
2. i1, i2, … i64: Sixty four binary inputs that are used to read in the clauses of a CNF formula as follows:
   The formula is read in one clause at a time. Reading in each clause takes two clock cycles, i.e. a sequence of two vectors of values for i1, … i64 is needed to read in a clause. Note that for each j in 1, … 64, there are three possibilities for the boolean variable aj: either the clause has aj, or it has aj or it has neither. We will say that aj is present in the clause if ij is is 1 in the first cycle and 0 in the next cycle. We will say ajis present in the clause if ij is 0 in the first cycle and 1 in the next cycle. We will say neither aj nor ajis present in the clause if ij is 0 in both the first and second

cycles. We never expect ij to be 1 in both the cycles, i.e. you can treat this as an input don't care. To signify the end of clauses to be read in, the Load signal is held low for 2 clock cycles, and is assumed to be held low until the circuit completes its calculation and presents the results at its outputs (read further below).

As an example, suppose I want to read in the CNF formula (a1 + a2) . (a1 + a2 + a3) . (a3 + a2). This would require 8 clock cycles, with 2 clock cycles needed to read in each of the 3 clauses, and Load being held 0 for the last 2 cycles (to denote end of the clauses). The following gives the vector of values of i1 i2 … i64 in each of the 6 clock cycles while Load is held at 1.

Clock cycle	**Load**	**i1**	**i2**	**i3**	**i4**	**...**	**...**	**i64**
1	**1**	**1**	**0**	**0**	**0**	**0...0**	**0...0**	**0**
2	**1**	**0**	**1**	**0**	**0**	**0...0**	**0...0**	**0**
		a1	**a2**	**Read clause 1**				
3	**1**	**0**	**1**	**1**	**0**	**0...0**	**0...0**	**0**
4	**1**	**1**	**0**	**0**	**0**	**0...0**	**0...0**	**0**
		a1	**a2**	**a3**	**Read clause 2**			
5	**1**	**0**	**0**	**1**	**0**	**0...0**	**0...0**	**0**
6	**1**	**0**	**1**	**0**	**0**	**0...0**	**0...0**	**0**
			a2	**a3**	**Read clause 3**			
7	**0**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
8	**0**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
		End of clauses						

3. The circuit is assumed to have 65 outputs, names SAT, a1, a2, … a64. At the end of its computation, the circuit should hold SAT at 1 for two clock cycles if the input formula read in is satisfiable. In addition, the values of a1, … a64 should give a satisfying assignment of the input formula read in. The circuit should hold SAT at 0 for two clock cycles if the input formula read in is unsatisfiable. The values of a1, … a64 do not matter in this case, i.e. they are output don't cares.

You are free to look up different techniques for solving the satisfiability of Boolean formulas for solving this problem. Note that a brute force search of trying all possible combinations (2^64 if all 64 variables are used in the formula) is not going to work in practice. 2^64 is approximately 10^20. Even if you took a clock running at a speed of 1THz (10^12 Hz), this would take 10^8 seconds, which is approximately 3.17 years! So you need to be clever about solving this problem. While in the worst-case, your circuit might require a very long time to solve this problem, in most cases, it should be able to solve it relatively quickly. Note that modern SAT solvers (implementedin software) can check satisfiability of a large class of Boolean formulas containing 100s of thousands of variables in a matter of a few hours on a desktop computer (running at say 3GHz).

You should pay attention to the time taken by your circuit to solve the problem, and also the total amount of hardware used by your circuit. You get bonus points for using as less hardware as possible, while obtaining reasonable performance checking satisfiability of 64-input functions.

You can look at how software SAT solvers like picoSAT and miniSAT work from the internet. You can also look at http://www.satlive.org for a treasure trove of information about SAT solving.

Streaming colour interpolater in a digital image:
A 100 x 100 resolution image with 16-bit encoding of each of Red, Green and Blue colours at each pixel is to be smoothened using an interpolation algorithm. The interpolation algorithm is as
follows:

Let r_{x,y}, g_{x,y} and b_{x,y} be the 16-bit unsigned integer RGB values of the pixel at coordinates x and y. We wish to modify the RGB values as follows:

r_{x,y} = A. r_{x,y} + (1-A). (r_{x+1,y} + r_{x-1,y} + r_{x,y+1} + r_{x,y-1})
g_{x,y} = B. g_{x,y} + (1-B). (g_{x+1,y} + g_{x-1,y} + g_{x,y+1} + g_{x,y-1})
b_{x,y} = C. b_{x,y} + (1-C). (b_{x+1,y} + b_{x-1,y} + b_{x,y+1} + b_{x,y-1})

The values of A, B and C are parameters of the interpolation algorithm, and are obtained by dividing the values obtained from three 16-bit inputs iA, iB and iC by 2^16. Thus, if the inputs iA, iB and iC are respectively 1000, 10000 and 20000, then the values of A, B and C to be used in
the above interpolation algorithm are 1000/2^16, 10000/2^16 and 20000/2^16 respectively.

The values of r_{x,y}, g_{x,y} and b_{x,y} obtained from the above interpolation algorithm must be **rounded to the nearest unsigned integer.**

You are required to design a streaming interpolating circuit to implement the above interpolation algorithm. The values of r_{x,y}, g_{x,y} and b_{x.y} in the original image are fed as streams on
three 16-bit inputs of the circuit. The order of x,y in which these values are fed in is as follows:

The inputs start coming from pixel 0,0, and then continue as 0,1, all the way up to 0, 99. The next values come from pixel 1, 99, and then continue as 1, 98 all the way down to 1,0. The next values come from pixel 2,0, and then continue as 2,1, all the way up to 2, 99.

In other words, the values of x and y change as follows:

Start from x=0, y=0.
Set yIncreasingFlag = 1

repeat until (x = 99 and y = 0)
If (yIncreasingFlag = 1) AND (y < 99), the next value of x and y are the same as x and y+1.
If (yIncreasingFlag = 1) AND (y = 99), set yIncreasingFlag = 0, and the next values of x and y
are the same as x+1 and 99.
If (yIncreasingFlag = 0) AND (y > 0), the next value of x and y are the same as x and y-1.
If (yIncreasingFlag = 0) AND (y = 0), set yIncreasingFlag = 1, and the next values of x and y are
the same as x+1 and 0.
For boundary cases, i.e. values of x where x+1 or x-1 does not lie in the range 0 through 99, and for values of y where y+1 or y-1 does not lie in the range 0 through 99, you may assume that the RGB values for "out-of-boundary" pixels are all 0's.

Your circuit will be provided three 16-bit unsigned integers representing the RGB values of a pixel at x,y at each clock cycle. The sequence in which the pixel values are fed in is as given above. Your circuit should have three 16-bit unsigned integers as outputs representing the interpolated RGB values of the pixels in the same sequence in which the values are fed in. However, there could be some delay between the first pixel values coming in and the first pixel values going out.
Your circuit should have an OutputReady output that is held at 1 once the interpolated RGB values start streaming out of the circuit's outputs.

Your circuit should have a Reset input and a Start input. By holding the Reset at 1 for two clock
cycles, your circuit should get reset, i.e. ready to start processing a stream of pixels. Reset must then be held at 0 while the interpolation computation is ongoing in your circuit. The Start input is
held at 1 for two clock cycles, while Reset is 0, before the first pixel values are fed into the inputs.
For the next 100x100, i.e. 10000 clock cycles, the values of the pixels are fed in the sequence described above. The OutputReady output of the circuit should be held at 0 until the first interpolated RGB value is ready to be output. Subsequently, in every clock cycle, the RGB outputs should give the interpolated RGB values of the pixels in the same sequence in which the input pixels were read in.

You should pay attention to both the time taken and the hardware resources used in your project.