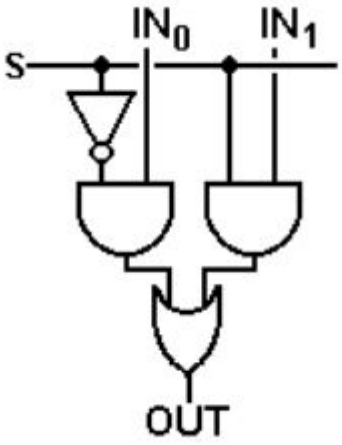
Documentation
CO225
Lab 05-Part 05
Group 21(E/17/219,E/17/027)

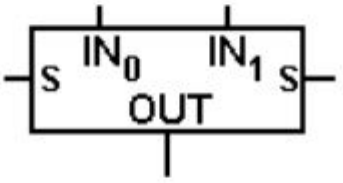
1.Left shift
-OP Code:1001
-Timing delay:1 unit
-Timing Diagram

PC Update	Instruction Memory Read		Register Read	ALU	
#1	#2		#2	#1	
	PC+4 Adder		Decode		
	#1		#1		
Register Write					
#1					

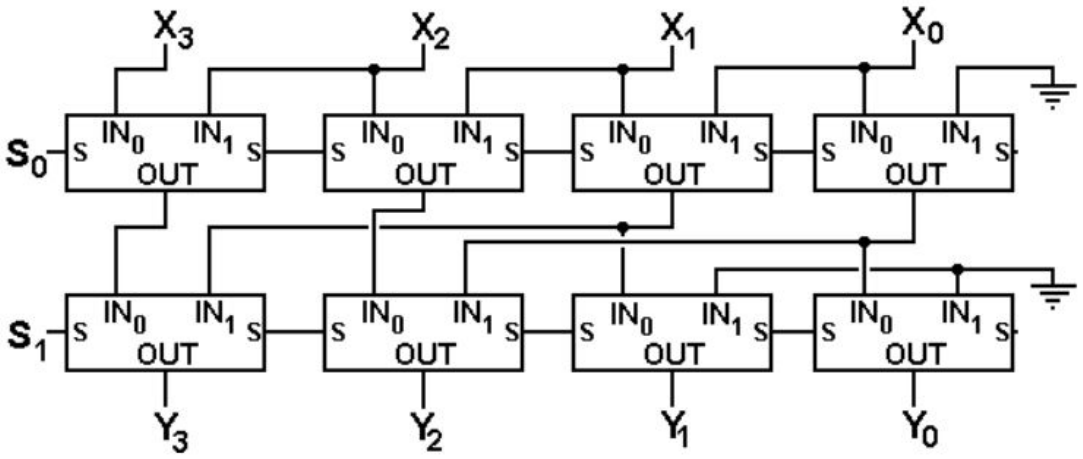
-Implemented the left shifter using gate level modeling.(block diagram is shown below.)

-leaf module





-top level design



--Note that this design is for 4 bit data input.In our implementation,this design is extended to support 8 bit input.

--functionality of this design

Signals		Action	Top Unit	Bottom Unit
$S_1 = 0$	$S_0 = 0$	No shift	No shift	No shift
$S_1 = 0$	$S_0 = 1$	Left shift one place	Shift by 1	No shift
$S_1 = 1$	$S_0 = 0$	Left shift two places	No shift	Shift by 2
$S_1 = 1$	$S_0 = 1$	Left shift three places	Shift by 1	Shift by 2

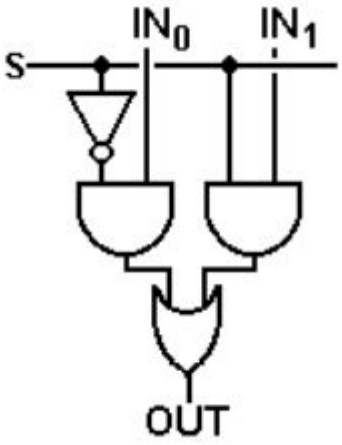
-Implementation

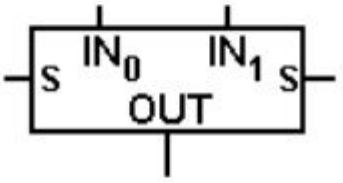
-First left shifting module was defined as in the LEFTSHIFTER.v file. Then it is thoroughly checked with the test bench which is inside the same file.After that inside the ALU the left shifting module was instantiated.Then the ALUOP new case was added so that if the ALUOP signal from the control unit is 4 then the left shifting was selected as the output of the ALU.Then the control signal is edited.If the OPCODE is 9 then the ALUOP signal is set to 4.

2.Logical right shift,Arithmetic right shift,Rotation
-All of these instructions are implemented using only one module.
-OP Codes:1010,1011,1100
-Timing delay:1 unit
-Timing Diagram for all three instructions

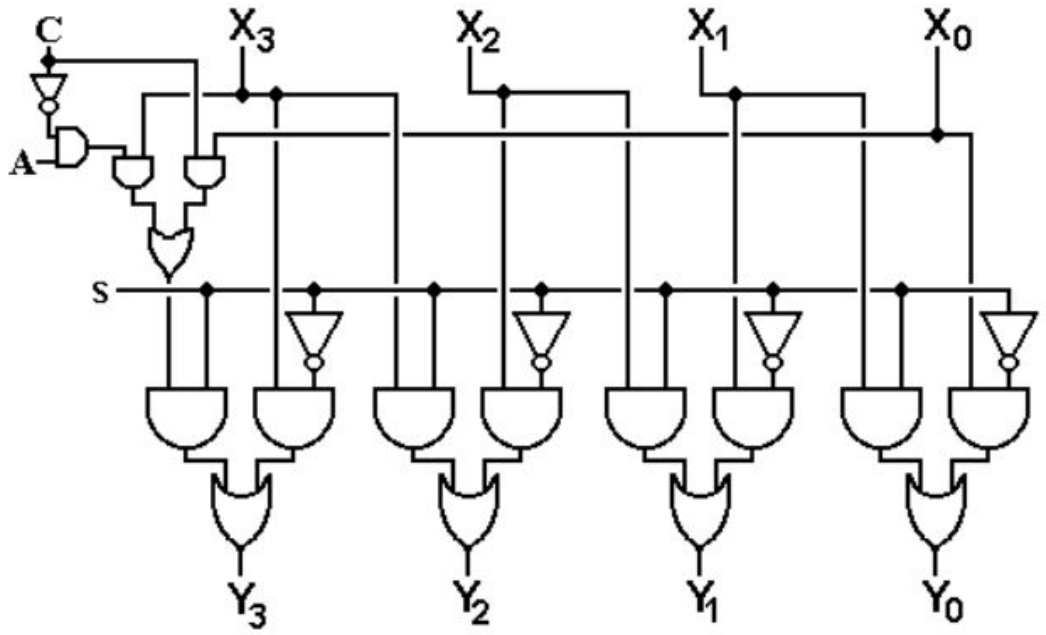
PC Update	Instruction Memory Read		Register Read	ALU	
#1	#2		#2	#1	
	PC+4 Adder		Decode		
	#1		#1		
Register Write					
#1					

-Implemented the left shifter using gate level modeling.(block diagram is shown below.)
-leaf module





-top level design



-Note that this design is for 4 bit data input.In our implementation,this design is extended to support 8 bit input.

--functionality of this design

A = d	C = 1	Circular right shift. This is an arbitrary decision, but the signals A and C should not be asserted simultaneously.
A = 1	C = 0	Arithmetic right shift
A = 0	C = 0	Logical right shift

-Implementation

-Logical right shifting , arithmetic right shifting and rotating all are done by using only one module.The desired behaviour is selected using different inputs.First the module was defined as in the RIGHTSHIFTER.v file. Then it is thoroughly checked with the test bench which is inside the same file.After that inside the ALU the three instances of this module was instantiated.Then in those three instances required three behaviours were achieved by giving different inputs to the A and C.Then by looking at the ALUOP signal it is decided which module is to use. After that control unit signal is edited.So that the required output can be selected using ALUOP signal.

3.BNE signal
-OP Code:1000
-Timing Diagram

PC Update	Instruction Memory Read		Register Read	2's Comp	ALU
#1	#2		#2	#1	#2
	PC+4 Adder		Branch/Jump Target Adder		
	#1		#2		
			Decode		
			#1		

-This was a simple implementation.
Control unit is edited so that the new signal called BNESIGNAL is generated.If the opcode is 1000 from the instruction, then the BNESIGNAL is set to 1 otherwise 0.

Then the Zero output was negated by using a not gate.After that that signal and BNE signal was anded using an and gate.Then the output of that and gate added to the previously defined or gate which is from the alu result.(Refer to the block diagram)