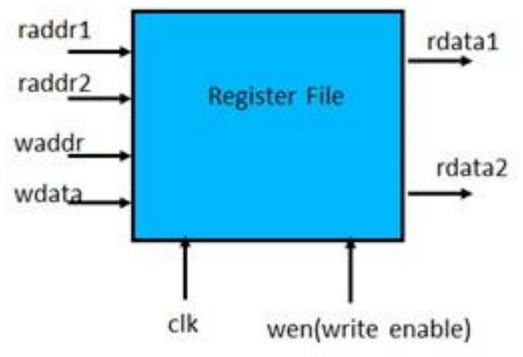
Faculty: Dr. Sharad Sinha
TA: Pavitra Bhade

In this lab you will analyse the implementation of the Register File (RF). Besides, you also need to find the area and access time of RF of different sizes (number of registers) and for various bit-widths of registers. You are provided with parametrizable Verilog code where the bit-width and the number of registers can be changed. You can specify the bit-width and the number of registers in the Verilog code of the RF and find the change in LUT consumption and access speed from the synthesis results.

I. PARAMETRIZABLE IMPLEMENTATION OF RF

You are provided with the behavioral design in Verilog for implementing the parametrizable RF where the bit-widths and number of registers can be changed to find the hardware and access time corresponding to different bit-widths and address-widths. The block diagram of the register file is given in Fig.1



The RF is positive edge triggered; which means that the data available at the write port (wdata) is written into the selected register in the register file on the positive edge of clock if the write enable (wen) signal is high. The register in which the wdata could be written is selected by a decoder according to the write address (waddr).The read ports are combinational. Data stored in the registers specified by raddr1 and raddr2 are obtained as radata1 and rdata2, respectively. All registers are reset to 0x0000 when the reset signal is **high**.

1. The Verilog file ‘regfile.v’ (provided) is a parametrizable behavioral code.
2. The ‘define.v’ file is used to define DSIZE (bit-width of the register) = 4, NREG (number of registers) =4, ASIZE (address size)=2. The testbench ‘regfiletest.v’ (provided) helps to test the behavioural implementation of RF. The test-bench uses file operation (an input file is used to read data therefrom and an output file to write data).
3. Verify the operation by simulation.
4. Synthesize the given behavioural code and plot the ‘number of slices’ used vs ‘No. of registers’ and ‘the minimum clock cycle period’ in ns vs ‘No. of registers’ (change the parameter for No. of registers {NREG= 16, 32, 64}). Here we can keep the bit-width of each register (DSIZE parameter) to be constant (for example DSIZE=32).

Table 1: LUT consumption and delay for behavioral register file vs bit-width

No. of Registers(NREG)	Bit-width of the register (DSIZE)	No of register slices used	No of LUT slices used	Minimum clock Period in ns
4	32			
8	32			
16	32			
32	32			
64	32			

5. Synthesize the given behavioral code and plot the number of slices used vs bit-width of the register (DSIZE) and delay in ns vs bit-width of the register (change the parameter for bit-width of the registers {DSIZE= 16, 32, 64}). <u>Here we can keep the number of registers (NREG parameter) to be constant (for example NREG=32) and thus ASIZE will be a constant =5.</u>

Table 2: LUT consumption and delay for behavioural register file Vs Number of registers

Bit-width of the register (DSIZE)	No. of registers (NREG)	No of register slices used	No of LUT slices used	minimum clock period in ns
4	32			
8	32			
16	32			
32	32			
64	32			

Task -I

1) Plot the graph, area in LUT slices (vs) No. of registers (NREG) as well as delay (vs) No. of registers (NREG) for the register file module for NREG = 16, 32, 64. Set DSIZE (bit-width of each register) =32.
2) Plot the graph, area in LUT slices (vs) bit-width of register (DSIZE) as well as delay (vs) bit-width of register (DSIZE) for the register file module for DSIZE = 16, 32, 64. Set NREG (number of registers) =32.

Task -II

ARITHMETIC LOGIC UNIT (ALU) SPECIFICATION

In this part of the Lab, we consider a simple ALU that performs the computation for eight arithmetic and logical operation. The seven operations are: ADD, SUB, AND, XOR, COM, MUL and ADDI as described in Table 1.

Table 1 - Description of ALU Operations

Instruction	**Equation**	**Operation**	**Description**
ADD	A+B	Addition	Addition of A and B, where both A and B are in 2's complement representation

SUB	A-B	Subtraction	Subtraction of A and B, where both A and B are in 2's complement representation
AND	A&B	Logical AND	Bit-wise AND of A, B
XOR	A^B	Logical XOR	Bit-wise XOR of A, B
COM	A << = B	Comparison	If A<=B then it results "true" (output 1) else "false (output=0) A and B are in 2's complement representation
MUL	A*B	Multiplication	Multiplication of A and B. A and B are in 2's complement representation
ADDI	Add Immediate operation as discussed in the lecture.		

A and B are the data input ports of the ALU to feed maximum of two operands to the ALU. The ALU has 3-bit control input to perform one out of the 7 possible instructions which the ALU can perform. The information is also listed in Table 2. The encoding of the 7 instructions is listed in Table 3.

Table 2 - Port List Specification. The bit-width to be varied from 8 bit to 64 bit.

Port Name	**Port Direction**	**Description**
A	Input	First operand
B	Input	Second operand
op	Input	Indicates the ALU about the operation to be performed
Out	Output	Output of the operation

Table 3 - ALU operation encoding

Operation	**'op' value**
ADD	000
SUB	001
AND	010
XOR	011
COM	100
MUL	101
ADDI	110

ALU Implementation, Testing and Analysis

For this assignment,

A. You are given the Verilog code as well as the test bench for ALU. You have to test whether the ALU gives correct results.

B. You need to set the input bit-width to 8, 16 and 32 (one by one) and find out the number of slices used. You need to plot area (vs) bit-width as well as delay (vs) bit-width for the ALU module. In FPGA you cannot find area directly so instead of area you can take number of slices, which would be considered proportional to the area.