International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 10 (2018) pp. 227-237

Received: 7 Oct, 2017; Accepted: 6 August, 2018; Publish: 27 August, 2018

Investigation of FPGA based 32-bit RISC-Modulation Processor

Joseph Anthony Prathap[1*], T.S.Anandhi[2], Nagarjuna Malladhi[3], V.Roja[4]

[1, 3, 4]Department of Electronics and Communication Engineering,
Vardhaman College of Engineering, Hyderabad, India
japtuhi1116@gmail.com

[2]Department of Electronic and Instrumentation Engineering,
Annamalai University, Chidambaram, India
ans.instrus@gmail.com

***Abstract*: This paper presents the design of a novel RISC-Modulation Processor using the Field Programmable Gate Array. Until now, the RISC processor is designed with operations like arithmetic and logical; shifting; rotating and comparing along with the instruction pipeline, which is the heart of RISC processor. Conventionally, the modulation techniques are developed separately as per the requirement. In this work, the RISC processor design is facilitated with the modulation techniques like pulse width modulation, pulse code modulation, pulse position modulation, quadrature amplitude modulation, sine wave generation and cosine wave generation so as to satisfy both computer and communication manipulations. The real time validation is performed by the implementation of the RISC-Modulation Processor using the Xilinx Spartan FPGA family devices. The power consumption and timing performance of the RMP design are evaluated for the proposed method and compared with different FPGA implementations.**

***Keywords*:** RISC Processor, Communication Processor, Field Programmable Gate Array

I. Introduction

RISC processor is a processor which performs all operations including arithmetic and logical, shifting and rotating, read and write by using simple set of instructions. Basically, in RISC processor design, the concept of pipelining is included. The RISC processor along with the pipeline concept helps in the efficient performance of the processor. "Pipelining" as the name specifies arranges the instructions in pipe or sequence to pass through the different manipulation phases. In other words, the CPU of the RISC processor is instruction pipelining. Also the instruction throughput is increased by the use of the pipelining.

Without pipeline, all instruction execute with respect to the clock pulse. The drawback of the clock pulse based operation is that only one operation could be performed at any time. To overcome this, the instruction pipelining is included with the RISC. The pipelining allows five operations within the single clock pulse. The pipeline involves five stages of operations. They are stated as a) Instruction Fetch (IF). b) Instruction Decode (ID). c) Execute (EX). d) Read/Write (RW) and e) Memory Access (MA).

Recently, the instruction pipeline is customized according to the requirement of the RISC applications. The top-down pipelined RISC design achieves maximum throughput of execution with less clock cycles per instruction [1]. The pipelined 8 bit RISC minimizes power by the clock gating technique [2]. But the 5-stage pipelined RISC developed using the ARM processor consumes more power [3]. To overcome the power consumption of the pipelined instructions and its latency, the reservation station based on Tomasulo algorithm is designed [4].

The design of the RISC processor using the Field Programmable Gate Array (FPGA) device has evolved effectively due to the reconfigurable property, reliability, upgradability, high performance, less power, less area, and parallelism in hardware which helps to generate high resolution output and complex computation throughput [5][6]. The power estimation could be performed for each of the RISC processor units and related the FPGA based power with the manufacturing power technology gap [7]. The external memory interfaced with the RISC through the PIC controller uses the reconfigurable FPGA design [8].The 32-bit RISC processor with floating point unit works at high speed [9]. The design of the RISC with circular convolution has less area and power [10]. The FPGA based RISC processor is designed with QR decomposition using Gram-Schmidt Ortho-normalization method to accommodate 79 instructions [11]. For the sake of removing the set-up slack, the Engineering Change Order (ECO) is utilized in the FPGA based RISC processor design [12].The FPGA is a device which has thousands of logic gates (AND plane and OR plane) drives for high performance. Another aspect of FPGA is high execution speed measured in Million Instruction per Second (MIPS) and reduced delays. The Xilinx Spartan FPGA is of several types such as Spartan 3A DSP, 3E, 6E depending on the number of AND gate and OR gate planes, LUTs, flip-flops within the device.

This paper proposes the novel RISC-Modulation Processor (RMP) design which utilizes the VHDL coded pulse modulation techniques like Pulse Width Modulation (PWM), Pulse Code Modulation (PCM), Pulse Position Modulation (PPM), Quadrature Amplitude Modulation (QAM) and signal generations (sine and cosine) with the RISC processor design. The combining of the RISC architecture with the DSP system finds advantage in many real time applications [13]. The next section presents the details of the proposed RMP design.

II. The proposed RISC-Modulation Processor

The RISC-Modulation Processor (RMP) is a combination of the RISC processor and the digital modulation techniques. Fig.1 shows the design flow for the proposed RMP. The RMP utilizes the pipelining concept for its operations. The instructions are stored in the memory in 32 bits. Depending on the instructions, the pipeline operation is initiated. The pipeline reduces the number of cycles while executing the instruction along with Read/Write function. The pipeline operation includes five sequences of operations as Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Read/Write Memory (RW) and Memory Access (MA).

The instruction has to be fetched from the memory and decoded or separated as per the instruction given. The 32 bits are split as opcode, input, output and RW operations. In this design, the RMP is included with 32 operations. Thus the opcode used in the RMP design is 2^5 bits. The operations in the RMP design is categorized as Arithmetic and Logical Unit (ALU), Shifting Unit (SU), Rotation Unit (RU), Comparator Unit (CU) and Modulation Unit (MU). The other important unit in the RMP design is the Control Unit (CTU) that performs control in pipeline operations, read/write operations, memory access operations and Program Counter (PC) operations. The RMP design is coded using the mixed style of the modeling in VHDL language. The following section discuss about the instruction pipeline in the proposed RMP design.

III. Instruction Pipeline in RMP using FPGA

A. Instruction Fetch (IF)

During the instruction fetch process of the pipeline, the 32 bit instruction is fetched from the memory. The IF operation uses the PC to keep track of the addresses for all the operations and increments by 1 for the next Instruction. The VHDL code for the IF requires the 32 bit register declared as array variable stored with the addresses of all the operations in the RMP design. The PC is developed using the increment design to track the current operation and predict the next operations in the RMP design.

B. Instruction Decode (ID)

The ID is the process of the splitting the 32 bit instruction to control according to the operation such as read/write and the number of inputs & outputs. For all operations, the VHDL code for the ID is developed so as to split the 32 bits instruction as opcode bits (4-0 bits) and the MSB namely 31st bit as Read/Write operation indicator. If the value of the 31st bit is '0', the read operation is activated and '1' value is used for the write operation activation. The bit splitting for the inputs and outputs depends on the operations being used. The ID for each of the operations used in the RMP is unique. The RMP design presents 32 different operations starting from the simple addition to the constellation for 64 bit QAM. The VHDL code for the 32 operations is developed in structural modeling, as the five units are instantiated as components. The preceding sections present each of the operations along with its instruction formats.

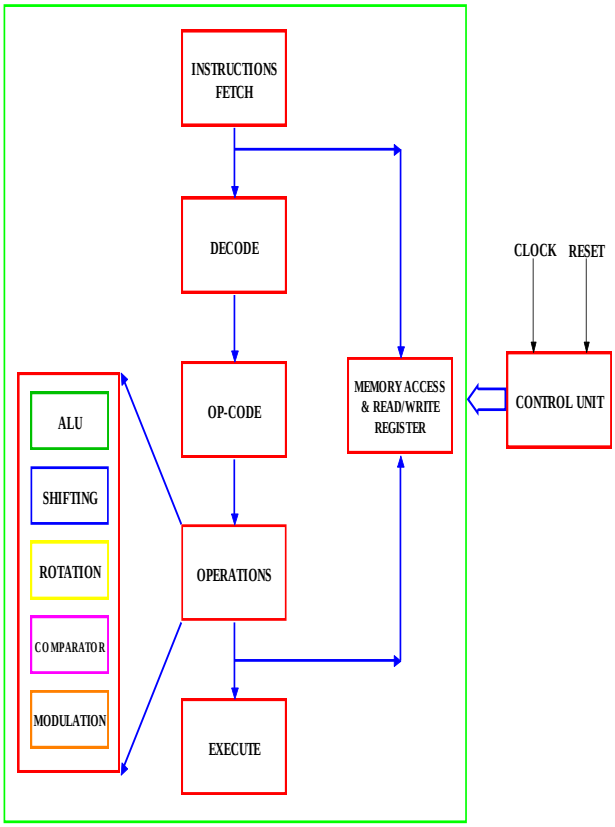


Figure 1. FPGA implemented RISC-Modulation processor

1) Arithmetic and Logical Unit (ALU)

The ALU consists of the arithmetic operations like addition, subtraction, multiplication and division. The logical operations are AND, OR, NOT, XOR, XNOR, NAND, NOR. All the arithmetic and logical operations are performed with respect to 2^8 bits, except the multiplication of two 8 bits number produce maximum 15 bits in the output due to the overflow. The increment and decrement operation is used for the PC unit to track and predict the addresses of the present and next instruction respectively. The instruction format for the ALU of the RMP is depicted in Table 1. The number of inputs utilized is indicated within the parentheses.

The VHDL code split the 32 bits instruction as per the operation required within the ALU. The bit split code is same for ALU operations except the multiplication, NOT logic operation, increment and decrement operations. Due to the prevalence of the output with 15 bits in multiplication operations, the multiplication operation is performed with the same two numbers. In other words, no two different 8 bits number are multiplied. The NOT logic is a single operator performed only for 8 bits of input. Increment and Decrement requires one bit for its operations

ALU OPERATION	RW	BITS FORMAT FOR INPUTS & OUTPUTS			OP--CODE
		OUTPUT	SECOND INPUT	FIRST INPUT	
Addition	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Subtraction	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Multiplication	31	27-13 (15)	--	12-5 (8)	4-0
Division	31	28-26 (3)	25-21 (5)	13-5 (8)	4-0
And Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Or Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Nand Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Nor Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Xor Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Xnor Logic	31	28-21 (8)	20-13 (8)	12-5 (8)	4-0
Not A	31	28-21 (8)	--	12-5 (8)	4-0
Not B	31	28-21 (8)	--	12-5 (8)	4-0
Increment	31	7 (1)	--	5 (1)	4-0
Decrement	31	7 (1)	--	5 (1)	4-0

Table 1 Instruction Format for the ALU of the RMP design

2) Shift Register Unit (SU)

The SU can also be referred as universal shift register unit. The SU consists of four types of registers implemented using the same hardware components that is controlled by the opcode. The four types of shift registers are Serial In Serial Out (SISO), Serial In Parallel Out (SIPO), Parallel In Serial Out (PISO), Parallel In Parallel Out (PIPO). The choice of the shift operation is decided by the opcode fed within the instruction set. The decode of the instruction fetched is performed for the SU of the RMP is as given in the Table 2

OPERATION	RW	BITS FORMAT FOR INPUTS & OUTPUTS			OP CODE
		LOAD ENABLE	OUTPUT	INPUT	
SISO	31	--	6 (1)	5 (1)	4-0
SIPO	31	--	13-6 (8)	5 (1)	4-0
PISO	31	16 (1)	15 (1)	12-5 (8)	4-0
PIPO	31	--	22-15 (8)	12-5 (8)	4-0

Table 2 Instruction Format for the ALU of the RMP design

The VHDL code for the universal SU is developed considering the instruction format. The input format is split as 1 bit for SISO & SIPO and 8 bits for PISO and PIPO. The Load Enable (LE) is used with the PISO operation, to transfer the parallel bit to the serial bit output. The behavioral model of the VHDL code is developed for the PISO.

3) Rotation Unit (RU)

The RU is performed by the barrel shifting. The sequence of 8 bits is called as barrel. The barrel shifting refers to moving specified number of bits out of the 8 bits input towards the given particular direction. In order to indicate the rotating direction, 1 bit is provided and for the number of bits to be rotated, 3 bits are used. The RU can be manipulated for the range of 1 to 5 bits of rotation in either direction. The VHDL code for the RU performs the splitting of the 32 bit instruction as shown in the Table 3. The number of bits for rotation and direction of the rotation is performed accordingly within the VHDL code. The behavioral style of coding is preferred for RU.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT				OP CODE
		OUTPUT	BITS	DIRE--CTION	INPUT	
ROTATION	31	28-21 (8)	17-15 (3)	14 (1)	12-5 (8)	4-0

Table 3 Instruction Format for the RU of the RMP design

4) Comparator Unit (CU)

The CU utilizes two 8 bit binary value comparisons for 3 conditions. The condition validation is performed for greater than, less than and equal to. The instruction decode for the CU as presented in Table 4. The decoded values of the inputs are 8 bits each along with the 5 bits opcode are coded by the VHDL code.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT			OP CODE
		OUTPUT	SECOND INPUT	FIRST INPUT	
EQUAL	31	28 (1)	20-13 (8)	12-5 (8)	4-0
GREATER	31	28 (1)	20-13 (8)	12-5 (8)	4-0
LESSER	31	28 (1)	20-13 (8)	12-5 (8)	4-0

Table 4 Instruction Format for the CU of the RMP design

5) Modulation Unit (MU)

The MU is developed using the mixed style of modeling in VHDL code to satisfy all the modulation techniques. The modulation techniques like PWM, PPM, PCM, QAM, sine wave and cosine wave generation are used in the MU. The instruction format for the all the modulation techniques in the MU is discussed below.

(a) Pulse Width Modulation (PWM)

The PWM is designed by comparing the high carrier wave with the DC signal. The duty-in value represents the DC signal equivalent in 2^8 bits. The carrier wave is the generation of triangle wave with the peak value as 255 for 2^8 bit resolution. The duty cycle is represented by 'D' and is given by

$$D = \frac{Ton}{Ton+Toff}$$

where T_{on} is the ON period and T_{off} is the OFF period.

The duty cycle is always represented in percentage (%). Depending on the DC value fed in, the pulse width changes. This RMP gets the instruction decoded as shown in Table 5. The VHDL code for the PWM generation involves in the carrier triangular wave generation and then comparing with the DC value. The PWM implemented using the FPGA is reliable and feasible [14]. The resolution of the PWM design is 2^8 bits.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT			OP CODE
		PWM 1	PWM 2	DUTY IN	
PWM	31	27 (1)	26 (1)	12-5 (8)	4-0

Table 5 Instruction Format for the PWM in the MU-RMP design

(b) Pulse Position Modulation

The PPM uses the sine modulating signal and overlaps with the carrier signal to produce the pulse output. The position of the pulse output prevails more with the low level of the modulating signal and as the level increases the position of the pulse output has some separation or gap between pulses. The resolution used is 2^8 bits. The enable signal is provided for the control of the PPM generation. The instruction format for the PPM is as given in Table 6. The VHDL code for the PPM generation is developed in the behavioral model which includes the enable signal.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT			OP CODE
		PPM	ENABLE	DUTY IN	
PPM	31	27 (1)	20 (1)	12-5 (8)	4-0

Table 6 Instruction Format for the PPM in the MU-RMP design

(c) Pulse Code Modulation

The discrete sine wave signal is subjected to two processes namely sampling and quantized to generate the steam of bits. The sampling process is the digitizing along the X-axis of the sine wave to obtain the sample sequences and quantization is the digitizing along the Y-axis of the sine wave to obtain the quantized levels. The resolution used for the PCM generation is 2^8 bit output for both sampling and quantization. The instruction fetched is decoded as given below in Table 7. The VHDL code for the PCM prefers the structural model of design. The VHDL code sine wave generation is functional input for the PCM circuit.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT		OPCODE
		PCM	SINE WAVE INPUT	
PCM	31	27-20 (8)	12-5 (8)	4-0

Table 7 Instruction Format for the PCM in the MU-RMP Design

(d) Quadrature Amplitude Modulation

The QAM is generated uses maximum of 64 bits. The 64 bit QAM requires 2^6 bits in which 2^3 MSB bits are used for the cosine wave and 2^3 LSB bits are used for the sine wave. The levels are assumed to be different as per the resolution given. For every value of the 2^6 bit, the QAM adds the corresponding levels of sine and cosine waves. Then the maximum magnitudes for each of the 2^6 bit values are taken along with its corresponding angles. The rectangle form of the obtained magnitude and angles are converted to constellation diagram for the QAM transmission. The instruction decode for the QAM is as below in Table 8. The VHDL code performs the above mentioned tasks using the mixed style of model.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT			OP CODE
		REAL PART OF QAM	IMAGINARY PART OF QAM	2^6 BIT INPUT	
QAM	31	27-18 (10)	17-8 (10)	10-5 (6)	4-0

Table 8 Instruction Format for the QAM in the MU-RMP design

(e) Cosine wave generation

The cosine wave generation is vital in the design of most digital and pulse modulation techniques. The cosine wave is generated to the maximum of 2^8 bits. The amplitude level of the cosine wave is varied by assigning values through the input. The amplitude of the cosine is limited to -127 to 127 for signed signals and 0 to 255 for unsigned signals. The VHDL code takes into considerations the both the signed and unsigned values for the amplitude levels of the cosine wave. Table 9 depicts the instruction format for the cosine signal generation. The resolution used for the cosine signal is 2^8 bits.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT		OPCODE
		COSINE SIGNAL	INPUT	
COSINE	31	17-10 (8)	9-5 (5)	4-0

Table 9 Instruction Format for the Cosine signal in the MU-RMP design

(f) Sine wave generation

Similar to the cosine wave, the sine wave is generated to the maximum of 2^8 bits resolution. The amplitude level limits to the -127 to 127 for signed and 0 to 255 for unsigned signals. The sine signal starts its value at '0'. The cosine wave starts at 90 ° phase shift of the sine wave. The instruction decode is given as in Table 10. The VHDL code is developed in the behavioral style of modeling for the sine wave generation.

OPERATION	RW	BIT FORMAT FOR INPUT & OUTPUT		OPCODE
		SINE SIGNAL	INPUT	
SINE	31	17-10 (8)	9-5 (5)	4-0

Table 10 Instruction Format for the Sine signal in the MU-RMP design

(g) Resolution

The resolution of the MU of the RMP design is 2^8 bits. The resolution is defined as the number of bits used for the design of modulations within the VHDL code. The modulation techniques like PWM, PPM, PCM, QAM, sine wave and cosine wave require precise resolution for accurate manipulation. Further increase in the resolution say $>2^8$ bits, increases the complexity of the proposed RMP method.

C. Execute (EX)

The EX is the stage at which the outputs are evaluated as per the operation given. The output of the ALU, SU, RU, CU and MU are of 2^8 bit length. The execution of the modulation block is the most challenging section of the RMP design. The methodology for each of the modulations is not related to any other modulation. The MU does not have common input for the modulation techniques within the unit. The EX is performed after IF and ID of the pipeline sequence. When any operation is executed at particular clock cycle, the IF & ID are performed for previous operations and RW & MA are manipulated for the preceding operations.

D. READ/WRITE (RW)

The 32 bit instructions when decoded get the information so as to read from the memory register or write the output to the memory register. The register length differs as per the operations used. The RW is an important operation for the RMP as it provides options to either store the results or retrieve the information of the previous output. The VHDL code for the read/write operations is designed using the behavioral style.

E. Memory Access (MA)

The MA is the final stage of the pipeline operation. In real practice, the MA is used at two phases of the pipeline process. The IF and RW utilize the MA for their performance. Depending on the type of RW operations, the memory is accessed at the two stages. IF uses the MA for READ operations. Similarly, the RW uses the MA for WRITE or READ operations.

OP CODE	OPERATIONS	OP CODE	OPERATIONS
00000	ADDITION	10000	EQUAL
00001	SUBTRACTION	10001	GREATER
00010	MULTIPLICATION	10010	LESSER
00011	DIVISION	10011	1 BIT L/R ROTATION
00100	AND	10100	2 BIT L/R ROTATION
00101	OR	10101	3 BIT L/R ROTATION
00110	NAND	10110	4 BIT L/R ROTATION
00111	NOR	10111	5 BIT L/R ROTATION
01000	XOR	11000	INCREMENT
01001	XNOR	11001	DECREMENT
01010	NOT A	11010	PWM
01011	NOT B	11011	PCM
01100	SISO	11100	QAM
01101	SIPO	11101	PPM
01110	PISO	11110	COSINE WAVE
01111	PIPO	11111	SINE WAVE

Table 11 Opcode with the operation description for the proposed RMP design

F. Control Unit (CU)

The Control Unit (CU) takes charge of the all the task in the RMP. The pipelining of the operations is organized according to the orientation of the instruction given. The operations of pipelining like IF, ID, EX, RW, MA are performed within one clock cycle. The RW along with MA requires control to read at the IF stage or RW stage. The enable signals are to be organized for few operations within the RMP units. The epode for all the 32 operations with description is given in the Table 11.

IV. Results and Discussion

The RMP design is executed in simulation using the MODELSIM software. The instruction of the RMP is fed such that all the operations pass through the pipeline process. Fig.2 shows the ALU operations with pipelining. The pipelining is only shown for the three operations of the ALU. The operations of addition, subtraction, multiplication, division and all logic operations are achieved by the ALU developed using the VHDL code. Fig.3 gives the shifting properties of the 8 bit register in the RMP. The SISO and SIPO use the 8 bit input in sequence. The PISO and PIPO use the 8 bit input in parallel.

Fig.4 shows the operational characteristics for the RU of the RMP design. The VHDL code for the versatile barrel shifter is developed for the rotation purpose. The versatile barrel shifter produces the rotation in both left and right direction, with the number of bits to be rotated. The ID transfers control to monitor the rotational direction by the number of specified bits as shown in Fig.4. The CU performs the equality, greater or lesser check operation for 8 bits using the proposed RMP. The simulation response for the CU is shown in Fig.5. The instruction decode for the CU specifies the operation through the opcode and the result of the comparator is stored in register memory as required. Fig.6 depicts the modulation operation of the RMP. As shown in the Fig.6, the outputs of the modulation techniques are converted into bits. The frequency of operation of the modulation techniques are varying as shown.

The RTL schematic of the proposed RMP design is depicted in Fig.7 indicating the functional blocks with its interconnections. Fig.8 presents the VHDL code for the proposed RMP in structural modeling consisting of ALU, shifting, rotate, comparator and modulation using Xilinx ISE 14.5. Table 12 shows the comparison of the proposed RMP with other RISC processor design using different FPGA devices say 3A DSP, Xilinx Spartan 3E and Virtex 6. The device utilization chart of the proposed RMP proves to be less in area compared to Ref [2]. Table 13 presents the timing analysis of the RMP design using the Xilinx Spartan 3A DSP, Xilinx Spartan 3E and Virtex 6 FPGA. The timing of the RMP design with respect to synthesis, mapping, placement and routing is low for the Xilinx Spartan 3A DSP FPGA implementation. Also the power and thermal estimation of the RMP design using the Xilinx Spartan 3A DSP, Xilinx Spartan 3E and Virtex 6 respectively are listed from Table 13. The power consumption for the proposed RMP using the Xilinx Spartan 3A DSP FPGA is as low as 0.115W.

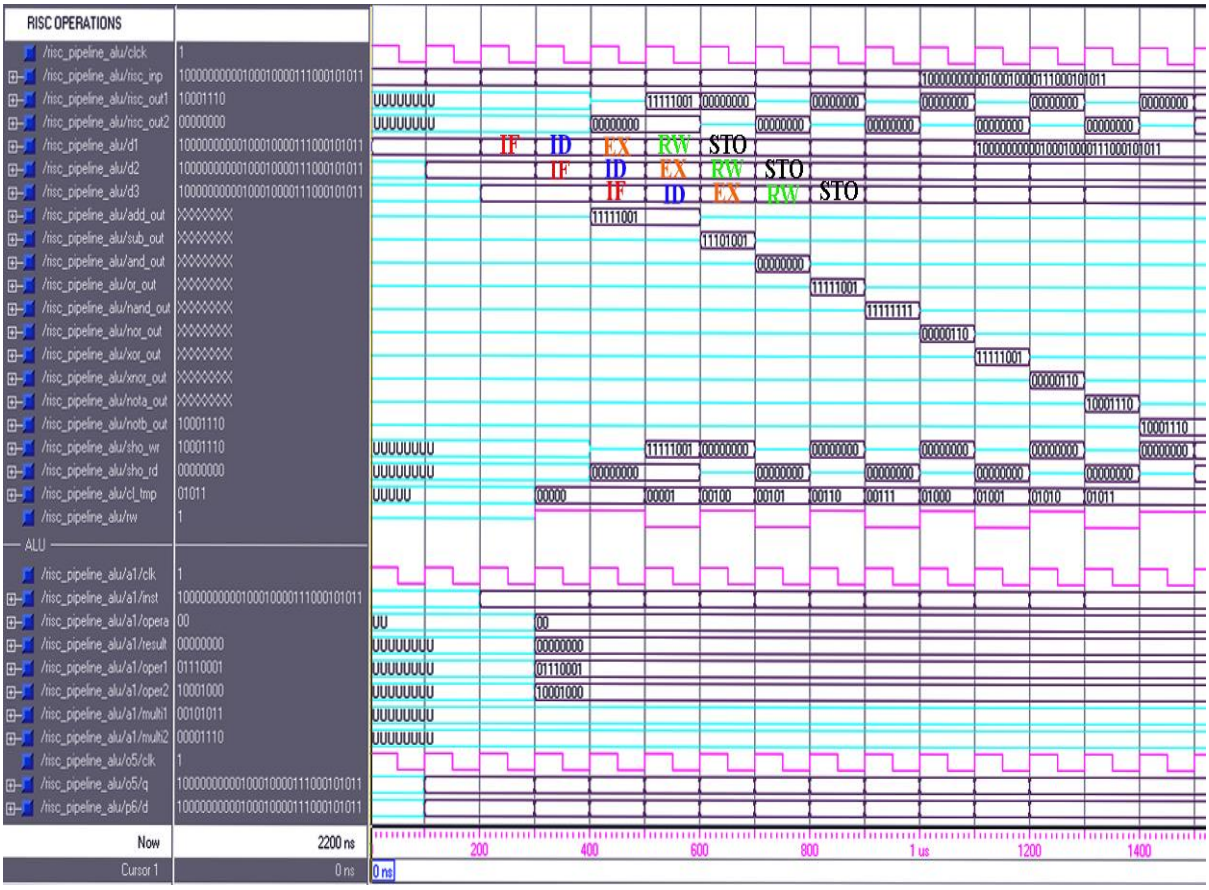


Figure 2. Simulation output for the ALU of the RMP design using MODELSIM

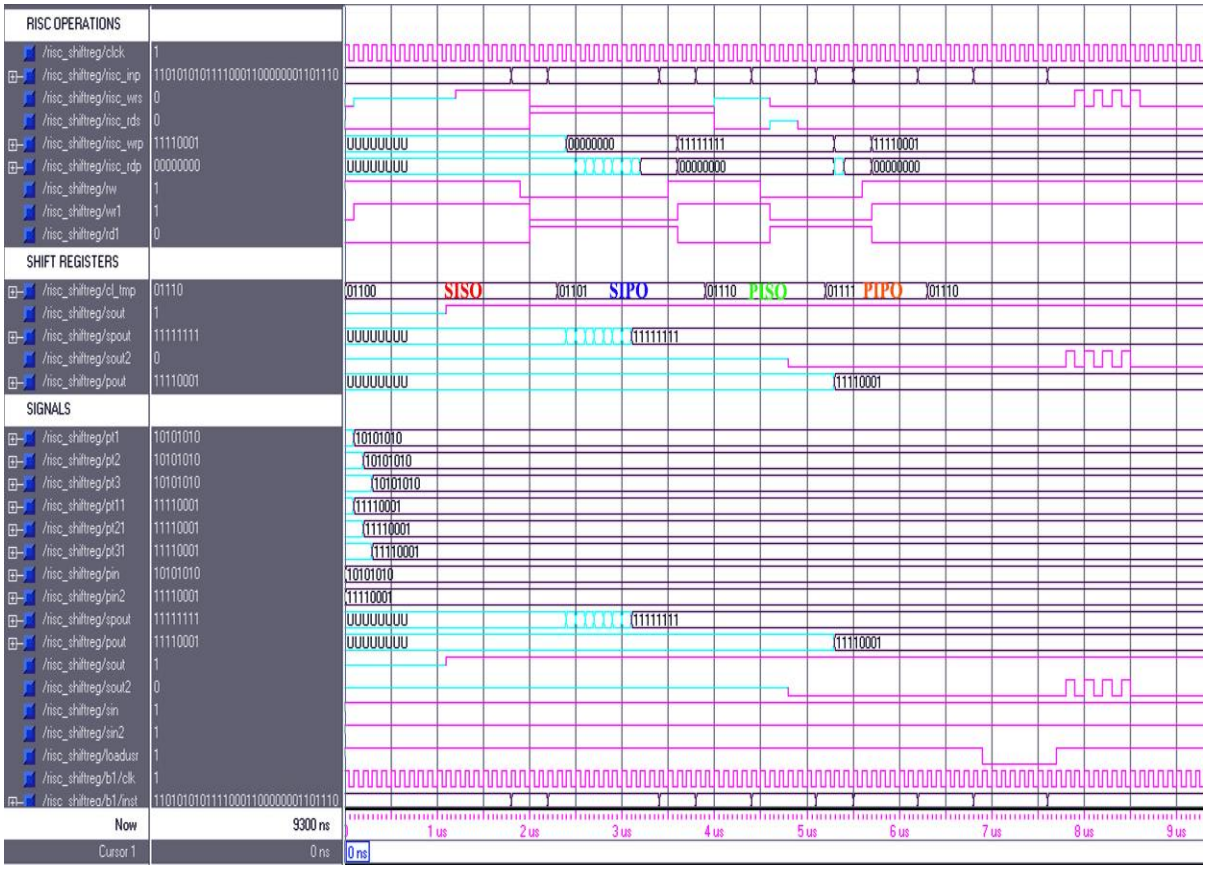


Figure 3. Simulation output for the SU of the RMP design using MODELSIM

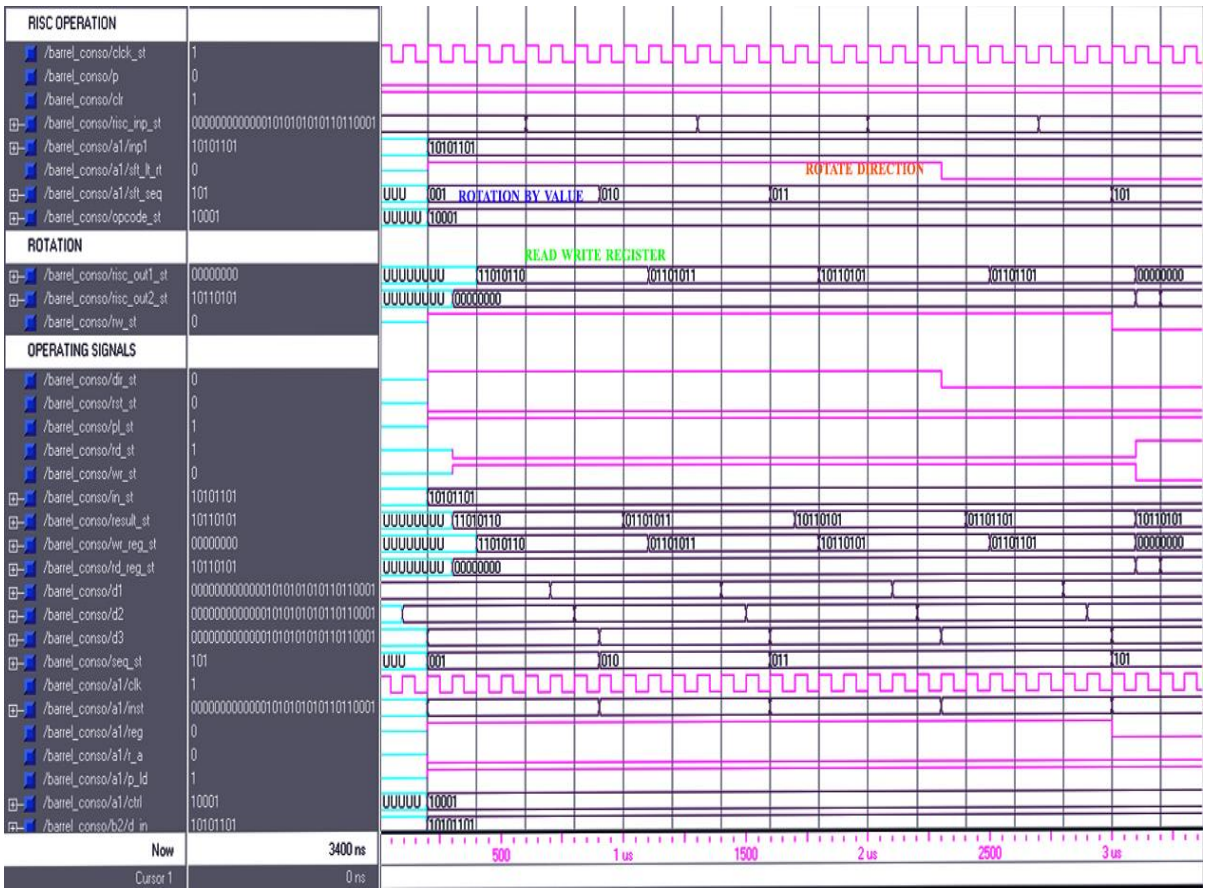


Figure 4. Simulation output for the RU of the RMP design using MODELSIM

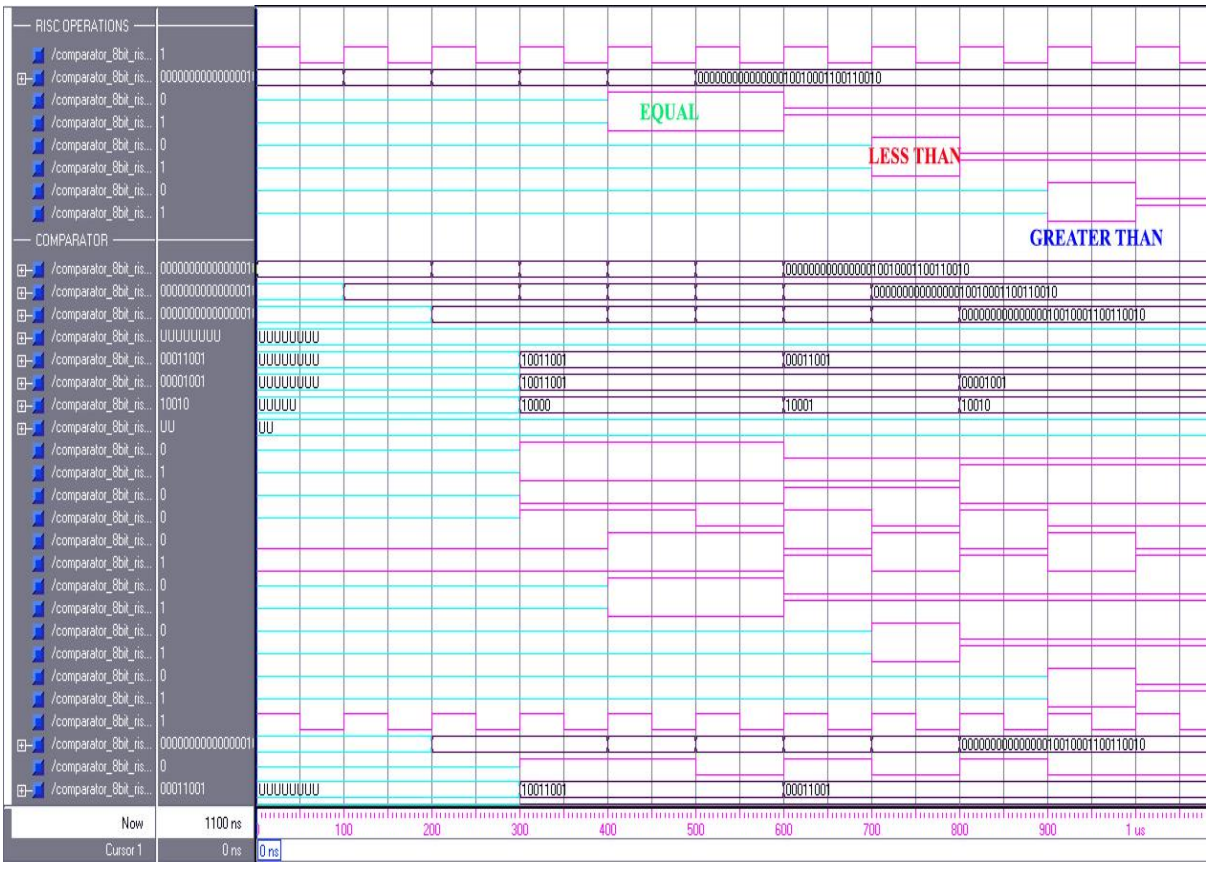


Figure 5. Simulation output for the CU of the RMP design using MODELSIM

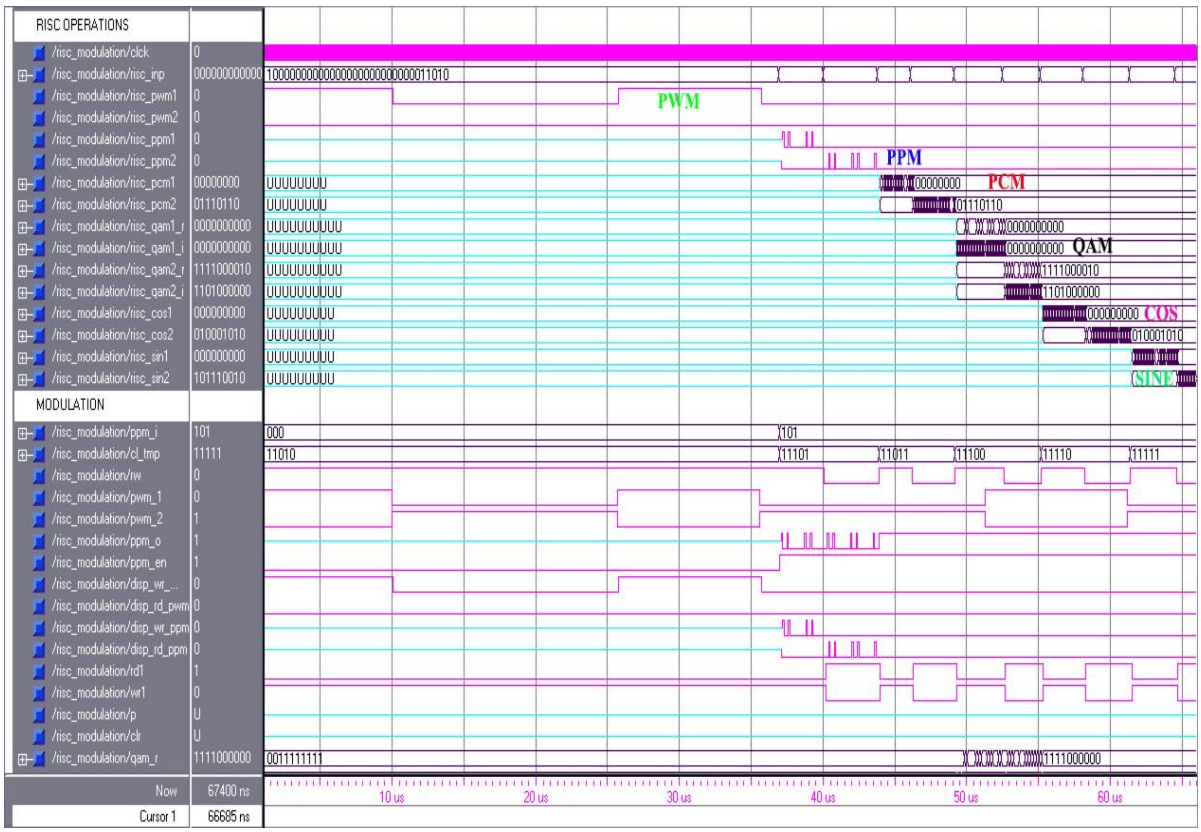


Figure 6. Simulation output for the MU of the RMP design using MODELSIM

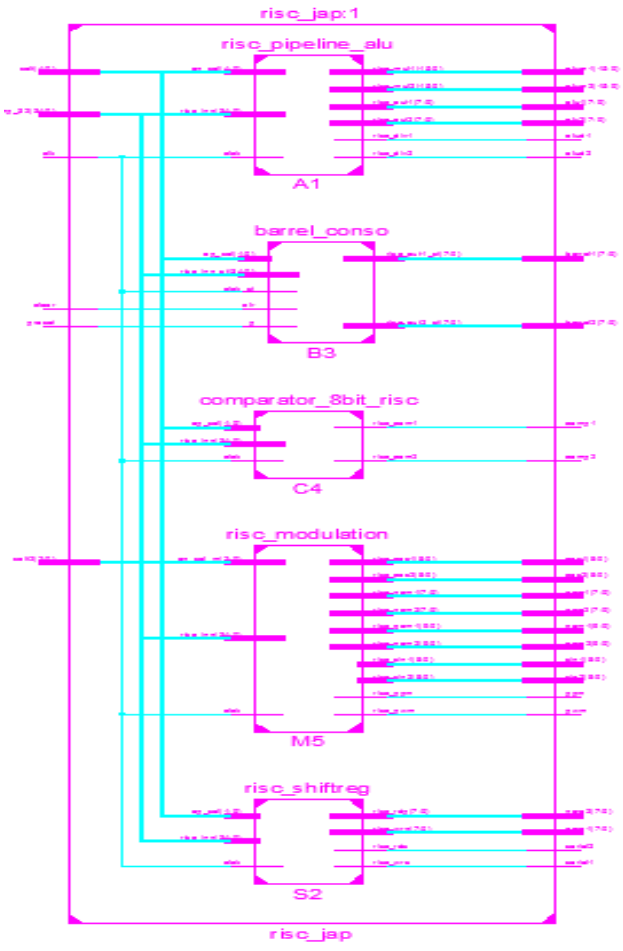
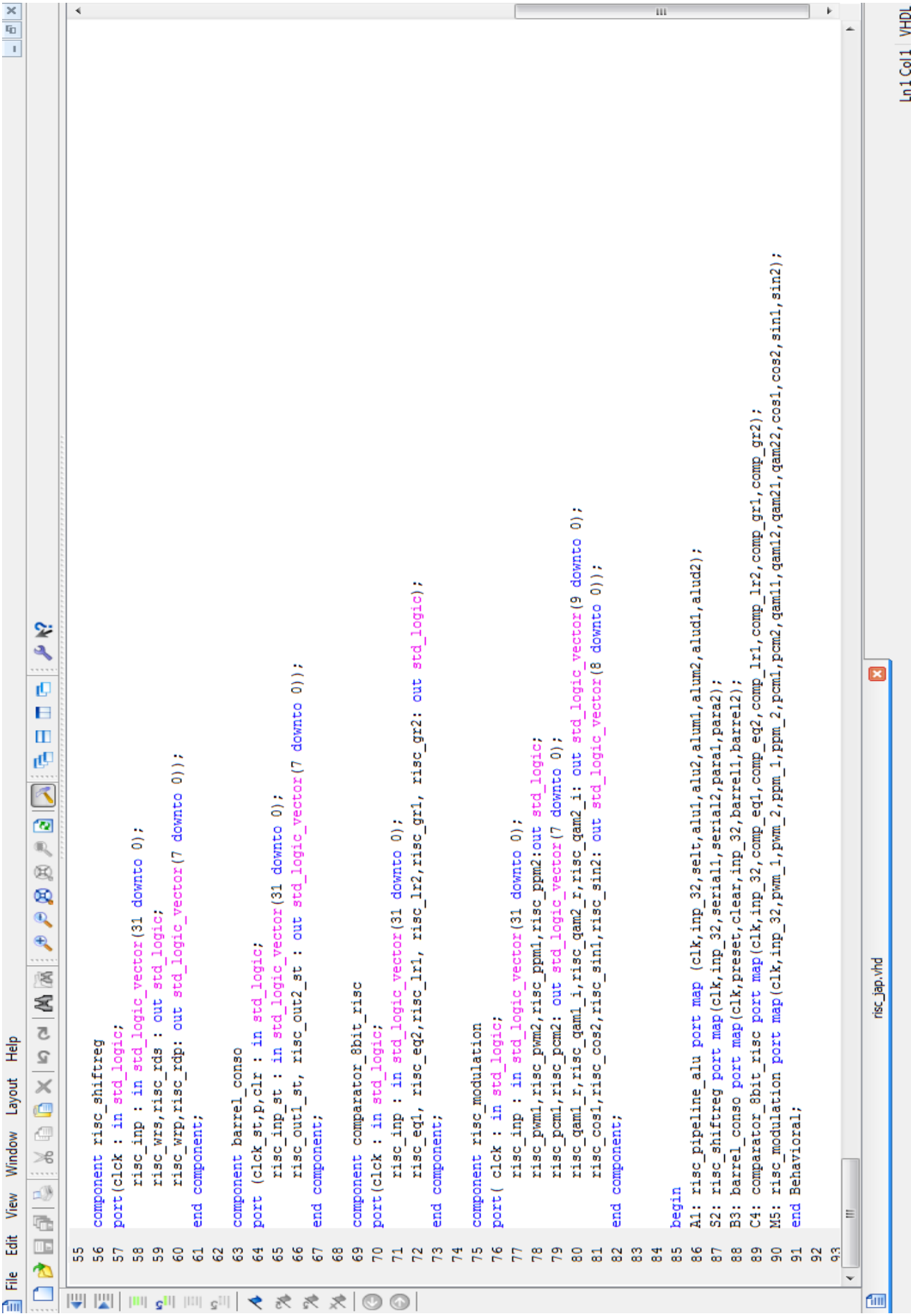


Figure 7. RTL view of the proposed RMP design using Xilinx Spartan 3A DSP FPGA



```
component risc_shiftreg
port(clck : in std_logic;
     risc_inp : in std_logic_vector(31 downto 0);
     risc_wrs,risc_rds : out std_logic;
     risc_wrp,risc_rdp: out std_logic_vector(7 downto 0));
end component;

component barrel_conso
port (clck_st,p,clr : in std_logic;
     risc_inp_st : in std_logic_vector(31 downto 0);
     risc_out1_st, risc_out2_st : out std_logic_vector(7 downto 0));
end component;

component comparator_8bit_risc
port(clck : in std_logic;
     risc_inp : in std_logic_vector(31 downto 0);
     risc_eq1, risc_eq2,risc_lr1, risc_lr2,risc_gr1, risc_gr2: out std_logic);
end component;

component risc_modulation
port( clck : in std_logic;
     risc_inp : in std_logic_vector(31 downto 0);
     risc_pwm1,risc_pwm2,risc_ppm1,risc_ppm2:out std_logic;
     risc_pcm1,risc_pcm2: out std_logic_vector(7 downto 0);
     risc_qam1_r,risc_qam1_i,risc_qam2_r,risc_qam2_i: out std_logic_vector(9 downto 0);
     risc_cos1,risc_cos2,risc_sin1,risc_sin2: out std_logic_vector(8 downto 0));
end component;

begin
A1: risc_pipeline_alu port map (clk,inp_32,selt,alu1,alu2,alum1,alum2,alud1,alud2);
S2: risc_shiftreg port map(clk,inp_32,serial1,serial2,para1,para2);
B3: barrel_conso port map(clk,preset,clear,inp_32,barrel1,barrel2);
C4: comparator_8bit_risc port map(clk,inp_32,comp_eq1,comp_eq2,comp_lr1,comp_lr2,comp_gr1,comp_gr2);
M5: risc_modulation port map(clk,inp_32,pwm_1,pwm_2,ppm_1,ppm_2,pcm1,pcm2,qam11,qam12,qam21,qam22,cos1,cos2,sin1,sin2);
end Behavioral;
```

Figure 8. VHDL code for the proposed RISC-Modulation Processor

Methods	Proposed RMP design			Ref[2]
	Spartan 3A DSP	Spartan 3E	Virtex 6	
Total Number Slice Registers	586	586	625	291
Number of 4 input LUTs	**761**	**763**	**655**	3374
Number of occupied Slices	**568**	**569**	**305**	1821
Total Number of 4 input LUTs	826	828	759	--
Number of bonded IOBs	223	223	223	166
Number of BUFGMUXs	1	1	2	--
Number of DSP48As	2	2	0	--
Average Fanout of Non-Clock Nets	3.00	3.01	3.61	--

Table 12 Device utilization comparison table for the proposed RMP design

Methods	Proposed Method		
	Spartan 3A DSP	Spartan 3E	Virtex 6
Max Delay	18.213ns	18.179ns	**7.170ns**
Number of paths	4533369	**4533261**	4537064
Number of destination ports	**722**	722	1123
Memory Utilized	216516 KB	**211396 KB**	223992 KB
Total REAL time to Xst completion	**19.00 sec**	19.00 sec	23.00 sec
Total Real Time to MAP	**8 sec**	10 sec	6 min 8 sec
Total Real Time to PAR	**3 min 57 sec**	7 min 15 sec	7 min 55 sec
Total Power (W)	**0.115**	0.203	4.357
Junction Temperature(℃)	**26.8**	29.3	55.2

Table 13 Timing analysis comparison of the RMP using three FPGA devices

V. Conclusion

The real time implementation of the RISC-Modulation Processor using the Xilinx Spartan 3A DSP FPGA found to be feasible and satisfactory. The power consumed and timing analysis for the proposed RMP design using Xilinx Spartan 3A DSP is low. The resolution used in the design is uniformly maintained as 2^8 bits for the purpose of simple circuit design. Future work could be directed towards advanced CISC-Modulation processor design using different resolution ranges from 2^8 to 2^{12} bits.

Acknowledgement

The authors would like to thank the department of Electronics and Communication Engineering, Vardhaman College of Engineering, Shamshabad, Hyderabad, India for the sustained encouragement and assistance provided for our successful completion of this work.

References

[1]. S. P. Ritpurkar, M. N. Thakare, G. D. Korde. "Design and Simulation of 32-Bit RISC Architecture Based on MIPS using VHDL". In *Proceedings of International Conference on Advanced Computing and Communication Systems*, 2015.

[2]. Jikku Jeemon. "Pipelined 8-bit RISC Processor Design using Verilog HDL on FPGA". In *Proceedings of the IEEE International Conference On Recent Trends In Electronics Information Communication Technology*, pp. 2023-2027, 2016.

[3]. Christiensen C. Arandilla, Joseph Bernard A. Constantino, Alvin Oliver M. Glova Anastacia P. Ballesil-Alvarez, Joy Alinda P. Reyes. "High-Level Implementation of the 5-Stage Pipelined ARM9TDM Core". In *Proceedings of the Technical Conference of IEEE Region10 (TENCON)*, pp. 1696-1700, 2010.

[4]. Nathaniel Albuquerque, Kritika Prakash, Anu Mehra, Nidhi Gaur. "Design and Implementation of Low Power Reservation Station of a 32-bit DLX-RISC processor". In *Proceedings of the International Conference on Information Science (ICIS)*, pp. 217-221, 2016.

[5]. Wang, W., Shen, Z., and Dinavahi,V. "Physics-Based Device-Level Power Electronic Circuit Hardware Emulation on FPGA". *IEEE Transactions on Industrial Informatics, 10(4)*, pp. 2166-2179, 2014.

[6]. Hwang, S.H., Liu, X., Kim, J.M., and Li, H. "Distributed Digital Control of Modular-based Solid-State Transformer Using DSP and FPGA". *IEEE Transactions on Industrial Electronics, 60(2)*, pp. 670-680, 2013.

[7]. Cosmin Cernazanu-Glavan, Marius Marcu, Alexandru Amaricai, Stefan Fedeac, Madalin Ghenea, Zheng Wang, Anupam Chattopadhyay. "Direct FPGA-based Power Profiling for a RISC Processor". *IEEE Instrumentation and Measurement Society*, 2015.

[8]. Suyash Toro, Sushma Wadar, Y. V. Chavan, S C Patil, D S Bormane, Avinash Patil. "External Memory Interface for RISC Controller on Reconfigurable Hardware Logic". In *Proceedings of the IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT)*, pp. 455-460, 2016.

[9]. Sangeeta Palekar, Nitin Narkhede. "32-bit RISC Processor with Floating Point Unit for DSP Applications". In *Proceedings of the IEEE International Conference on Recent Trends in Electronics Information Communication Technology*, pp. 2062-2066, 2016.

[10]. Suyog V Pande, Prashant D Bhirange. "An Efficient High Speed RISC Processor for Convolution". In *Proceedings of the IEEE sponsored 9th International Conference on Intelligent Systems and Control (ISCO)*, 2015.

[11]. Safaa S. Omran and Ahmed K. Abdul-abbas. "Design of 32-Bits RISC Processor for Hardware Efficient QR Decomposition". In *Proceedings of the International Conference on Advances in Sustainable Engineering and Applications (ICASEA)*, pp. 69-73, 2018.

[12]. Devaraconda Dinesh and R. Manoj Kumar. "Physical Design Implementation of 16 Bit Risc Processor". *Indian Journal of Science and Technology, 9(36)*, pp.1-7, 2016.

[13]. Amit Kumar Singh Tomar, Rita Jain. “20-Bit RISC and DSP System Design in an FPGA”. In *Proceedings of the Computing in Science & Engineering, Copublished by the IEEE CS and the AIP*, pp. 16-20, 2014.

[14]. Joseph Anthony Prathap, T. S. Anandhi and T. S. Sivakumaran. “Implementation of FPGA based DPWM-Digital PI Closed Loop Controller for Voltage Regulation”, *Indian Journal of Science and Technology*, 9(38), pp. 1-9, 2016.

Author Biographies

Dr. Joseph Anthony Prathap was born in 1981 in Puducherry. He has obtained B.E [Electronics and Communication] and M.Tech [VLSI Design] degrees in 2003 and 2007 respectively from Sathyabama University. He has put in 11 years of service in teaching and research. He is currently Associate Professor in the Department of Electronics and Communication Engineering at Vardhaman College of Engineering, shamshabad, Hyderabad, India. His research interest includes VLSI design, development of digital switch patterns, FPGA control techniques for power converters, photovoltaic power electronics converters.

Dr. T.S.Anandhi was born in 1974 in chidambaram. She has obtained B.E [Electronics and Instrumentation] an M.E [process control and Instrumentation] degrees in 1996 and 1998 respectively and then Ph.D in power electronics in 2008 from Annamalai University. She is currently Associate professor in the Department of Electronics and Instrumentation Engineering at Annamalai University, chidambaram, India and has put in 15 years of service. She has produced one Ph.D and guiding 6 Ph.D scholars. Her research interests are in power converters, control techniques for multiple connected power converters, embedded controllers for power converters, renewable energy based power converters. She is a life member of Indian society for technical education.

Mr. Nagarjuna Malladhi was born in 1988 in Andhra Pradesh**.** He has obtained B.E [Electronics and Communication] degree from Chirala Engineering College, Chirala in 2009 and M.Tech [VLSI Design] degree from Vignan University in 2011. He has put in 6 years of service in teaching and research. He is currently Assistant Professor in the Department of Electronics and Communication Engineering at Vardhaman College of Engineering, shamshabad, Hyderabad, India. His research interest includes VLSI design, Analog and Mixed signal.

Ms.V.Roja was born in 1992 in Telangana State**.** She has obtained B.Tech [Electronics and Communication] degree from Vijay College of Engineering for women, Nizambad in 2013 and M.Tech [Wireless and Mobile Communication] degree from JNTU, Hyderabad in 2016. She has put in 2 years of service in teaching. She is currently Assistant Professor in the Department of Electronics and Communication Engineering at Vardhaman College of Engineering, shamshabad, Hyderabad, India. Her research interest includes Wireless Communication, Mobile Networks and Sensors.