Chapter 5
Proposed Strategies for Power Optimization

5.1 Introduction

The proposed work suggested four out of many possible strategies; and all are implemented at hardware level for dynamic power reduction on the 32 – bit, 4- stage pipelined CPU whose architecture is discussed in previous Chapter, and the power comparisons are made with the conventional 5 – stage pipelined conventional CPU by taking the power reports for both the CPUs using Xilinx Xpower Analyzer. This chapter also discusses the 5- stage pipeline processor architecture in brief (for understanding purpose only) and using this processor the power comparisons are made.

With the prime focus to control the dynamic power along with maintaining the performance of the processor; we are implementing four strategies out of many other possible strategies upon the conventional 5 – stage CPU; which are (1) Memory stage Access Stage Removal, (2) Resource Sharing in 4 – stage CPU, (3) Novel RAM Addressing Scheme and (4) Clock Gating. After implementation of each technique the system is analysed for the power dissipation and at last the power comparisons with and without the implementation of these power reducing techniques on CPU are made and verified.

5.2 Architecture of 32 – bit 5 – Stage Pipeline (Conventional/Standard) CPU

Following Figure 5.1 describes the detailed architecture of 5- stage pipeline CPU, designed on the basis of RISC principle. This CPU is not described here in detailed as it is used only for the purpose of taking power results and then to apply the power reducing strategies to make it 4 – stage CPU which is discussed in Chapter 4. The power reports are generated using the Xpower Analyser (Xilinx ISE 13.1 Suit) for both the CPUs’ to compare the power
consumption and the performance, which will verify the proposed techniques for dynamic power reduction. Here, the conventional architecture of 5-stage CPU includes one more stage in addition to those of 4-stage CPU, which are Instruction Fetch (IF), Decode and Operand Fetch (DC), Execution (EX) and an additional one is Memory Access (MEM) and

Operand Fetch (DC), Execution (EX) and an additional one is Memory Access (MEM)
Write Back (WB). The memory access (MEM) stage is observed additional in 5 – stage CPU. For whole system the power analysis has been done and checked the power requirement using Xpower Analyser of Xilinx ISE – 13.1 suit, the following Figure 5.2 provides the details of power consumption of normal CPU with 5 – stage pipeline structure.

Figure 5.2: Summary of Power Consumption Report for 5 – Stage Pipeline CPU

The performance of above described 5 – stage CPU implementation has been verified for all the pipeline stages through the waveform generation using ModelSim6.5 and it can be represented as shown in following figures from Figure 5.3 to Figure 5.7. These simulated waveforms verifies the CPU performance in terms of execution of all types of instructions and the behaviour of all the related signals in all the pipeline stages.
Figure 5.3: Instruction Fetch for 5-Stage CPU
Figure 5.4: Instruction Decode for 5-Stage CPU
Figure 5.5: RAM Address for 5-stage CPU
Figure 5.6: Instruction Execute for 5-Stage CPU
Figure 5.7: Write Back Stage for 5 – Stage CPU
For the Instruction Fetch stage waveform depicted in Figure 5.3 shows the start of processor execution. And also shows that after the reset is de-asserted the program counter starts incrementing. In the Instruction Decode stage described by Figure 5.4, the instructions are decoded and corresponding signals are generated. As shown above waveforms the various relevant signals perform the operations.

The signals such as dependency signals (a_depen and b_depen) for handling hazards, branch signals to indicate a branch, register select for selecting registers, load and store signals for accessing block RAM, immed signal for indicating that current instruction is immediate, zerowrite and regwrite for indicating an update of a register after execution of instruction are involved in this stage.

Here in RAM Address stage in Figure 5.5 the Ram address signals are generated and fed to RAM, so that data can be made available in next clock cycle.

Instruction is executed during the Instruction Execute stage shown in Figure 5.6 and depending on types of instructions, either ALU operation is performed or RAM data loading or storing is done which is described in above waveforms. In Write Back stage detailed by Figure 5.7 the register bank of 16 registers is updated with the instructions.

5.3 Proposed Strategies for Power Reduction

5.3.1 Memory Access Stage Removal Technique

We are using Xilinx FPGA’s block RAMs to be utilised for RAM requirements of our system. Any write or storing of data to block RAM requires both address and data to be given at the same clock edge while data to be read or loaded from block RAM requires address to be given first and corresponding data will appear on the data line on the consecutive active clock edge. As data will be available on the next clock edge after issuing the address, one must keep a stage called memory access in the pipeline where the address is provided and data made available during the execute stage and should be stored in the corresponding register. This stage is available in 5-stage CPU.
Even though in 5 stage pipeline the memory access stage is available, it can be seen that this data memory access stage is not used by any of the arithmetic instructions or branch instructions. It is used only for memory access instructions. Arithmetic and branch instructions have a ‘NOP’ in the memory access stage. That is all its data are just passed-on through memory access and gets executed in the execute stage. Transitions during this unused state cause extra power dissipation. To avoid this wastage, the pipeline is reconfigured to bypass memory stage by using strategic address forwarding logic from instruction decode and operand fetch stage.

Thus address is provided to the RAM at instruction decode and operand fetch stage, if instruction is decoded to be the load instruction. Hence, if the data corresponding to that address is available in the execute stage and no extra clock or memory access stage is required. This address forwarding doesn’t affect the other instructions as RAM address generation logic is independent and does not directly or indirectly affect the logic of the consecutive or previous instruction in the pipeline.

For example consider add, load and sub instruction in the pipeline. Now while add instruction is getting executed, load instruction gets decoded and RAM address is provided. If RAM address is dependent on previous instruction the result generated of add instruction is directly given to RAM as address through MUX selection. Now if next sub instruction also has data dependency with the load instruction, then also there is no problem as data will be available in the execute stage which will be forwarded to sub instruction through data forwarding, implemented to remove data hazard. Similarly for any instruction in the pipeline there are no hazards with respect to address forwarding.

Above Figure 5.1 shows the architecture normal five pipeline stages where the memory access is required by the load instruction because address provided to read the memory in execute stage causes the read information to come in next clock cycle that is memory access stage. But for all other instructions these unused transitions do happen for memory access stage without affecting the normal operation; but these transitions during the unused stage cause extra power consumption. To avoid this wastage, the normal pipeline structure is required to modify during the design of architecture of CPU; in a way, so that the correct data forwarding takes place even for the load/store instructions when actually the memory access stage is required.
Thus, the normal 5 – stage pipeline structure is modified to 4 – stage pipeline structure to reduce the dynamic power consumption, whose detailed architecture has been explained in previous chapter in Figure 4.1 and power analysis has been done for this new architecture. It verifies the justifiable reduction in power dissipation. The summary of power analysis is shown in Figure 5.8.

Comparing the total power consumption requirement of the standard 5 – stage pipeline CPU, the 3mW (2.65%) improvement is achieved due to implementation of memory access stage removal technique.

Up on reduction of one pipeline stage, which was used for memory access, from the standard 5- stage pipeline, it is required to verify the performance of the 4 – stage CPU through the waveforms for all the stages. It is verified and noted that the power reduction is achieved without affecting the performance of the implementation. Explanations for waveforms for all the 4 – stages of CPU given through the Figure 5.9 to Figure 5.12 are same as that of for 5 – stages given above except the address forwarding is done from decode stage for RAM Access related instructions as discussed above.
Figure 5.9: Instruction Fetch for 4-Stage CPU
Figure 5.10: Instruction Decode and Operand Fetch for 4-Stage CPU
Figure 5.11: Instruction Execute for 4 – Stage CPU
Figure 5.12: Write Back for 4–Stage CPU
5.3.2 Resource Sharing Technique

The proposed technique is implemented during the designing of the decode stage of the processor. Most of the processors’ design supports two types of the instructions addressing modes which are immediate addressing and direct register addressing types. For example, the arithmetic operation addition can be performed by both the types of addressing modes as shown below:

Immediate Addressing Type Operation:  
Addi  r1 , 30 – bit Immediate Data

Direct Addressing Type Operation:  
Add  r1 , r2

Here, both the instructions performs almost the same operation i.e. addition where the operand 1 is common i.e. register r1. The operand 2 can be either immediate data value or a register depending on the type of instruction. To introduce resource sharing technique both the opcode is required to decode to same operation that is for addition Addi the opcode is 001110 and for Add the opcode is 000001 is decoded to perform the addition operation and assigned an opcode which is 000101.

Instead of using the separate resources for both the instructions which perform the same operation, this technique channelize the instruction into one and hence the saving of half the resources is achieved from the decode stage which would have been required if both the instructions were executed differently using their separate resources. Thus in execute stage only one adder does the work for both types of instructions. Thus much amount of power was saved by resource sharing just by adding some control signals in the decode stage and eliminating many flip – flops, comparators and muxes which would have been required in the later stage.

Through Resource sharing following instructions listed in Table 5.1 are channelized to one instruction and the considerable power saving was achieved. This technique can be applied to all the stages with different logic as applied to decode stage here.

<table>
<thead>
<tr>
<th>Instruction 1</th>
<th>Instruction 2</th>
<th>Operation</th>
<th>Decoded to Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add (000001)</td>
<td>Addi (001110)</td>
<td>Addition</td>
<td>Addition (000101)</td>
</tr>
</tbody>
</table>

Table 5.1: List of Instructions under Consideration
Thus from fetching, executing and write back stages considerable amount of resources are saved. During the instruction fetch operation the Program Counter (PC) is either incremented by 1 or incremented/decremented by the content of branch address from the current position of PC. To perform this operation two adders and one subtractor is required. During the design of processor 2’s complement logic was used for instruction fetching during the branching operation, which removes one subtractor.

Further, the resource sharing is also used by implementing 2:1 MUX with inputs of value 1 and 2’s complement branch address and a branch signal as select line. This MUX output is then added to current PC to generate the next PC value. This reduces further one adder as shown in following Figure 5.13.
The Program Counter operation can be expressed as follows:

\[ \text{PC} = \text{PC} + \text{PC}_{\text{incr}}; \]

In Execute Stage the ALU required 2 Adders for the operations such as addition and increment i.e. one for addition (Add, Addi) and one for Increment (Inc) and 2 subtractors for subtraction and decrement i.e. one for subtraction(Sub, Subi) and one for Decrement (Dec).

In this proposed work during execute stage instead of using 4 Adders/Subtractors dedicated to each operation one adder and a multiplexer have been implemented during the design of the CPU by which the resource utilization is optimized and the power reduction is achieved.

The design and the functionality of the newer logic which is implemented in the CPU can be explained through the following example and the Figure 5.14 explains the implementation logic.

Consider four operations which are to be executed:

(1) Add : \(a + b\); (2) Inc : \(a + 1\);
(3) Sub : \(a - b\); (4) Dec : \(a - 1\);

Figure 5.14: MUX with Opcode as Selection Logic
To achieve the resource sharing, B register is given the input value through the 4:1 MUX with the opcode as select line, which is as shown in Figure 5.14. It is concluded after the implementation of this logic that the single adder is used to generate the results and the logic used for all the cases can be presented as $C = a + b$ as operation.

After implementation of this logic and the overall power requirements are analysed for the modified 4- stage CPU and it is observed that 4mW (3.63%) further improvement in the total power consumption is achieved, the screen shot of the summary of power analysis is show in Figure 5.15.

![Figure 5.15: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of Resource Sharing Strategy)](image)

### 5.3.3 Novel RAM Addressing Scheme

Xilinx FPGA’s have asynchronous block RAM. We are using it as simple dual port RAM through coregen generator for this architecture which provides synchronous interface to this block RAM as dual port RAM. For write operation write_en is also generated with address and data is fed to the RAM, but for read, only address is provided which causes data to be
read from that memory location. Power will be dissipated for each new address fed to the RAM as it will require appropriate read or write to that memory location addressed by RAM. Thus we see that even though instruction is not a RAM related instruction, RAM address will be generated and corresponding data will be given out by RAM. But as instruction is not a RAM related instruction, this data will not be propagated to write back stage. Thus what was done in this power reduction strategy was giving a fixed address of x”7ff” to RAM when instruction was not a RAM access instruction. Thus the changes in the RAM address generation were MUXed out and as there was no unnecessary switching takes place the dynamic power reduction is achieved.

![RAM Address Multiplexer](image)

As seen in Figure 5.16 ram_address generated in the execute stage is fed to RAM, if there is load or store instruction else constant hexadecimal data 7FF is fed. Thus considerable amount of power is saved by preventing the occurrences of unnecessary switching by feeding constant address during execution of other instructions.

### 5.3.4 Clock Gating

Xilinx recommends for using the CLB clock enable pin instead of gated clocks. Gated clocks can cause glitches, increased clock delay, clock skew and other undesirable effects. Using clock enable saves clock resources and can improve timing characteristic and analysis of the design. But for the power reduction purpose if it is required to use a gated clock, most of the FPGA devices are facilitates with a clock enabled global buffer resource called BUFGCE.
However, a clock enable is still a better and preferred method to reduce or stop the clock to reach to various portions of the design and hence, subsequently to reduce the power.

There are several ways to use clock-enable resources which are available on devices, but to gate entire clock domains for power reduction purpose; it is preferable to use the clock-enabled global buffer resource called BUFGCE shown in Figure 5.17.

![Figure 5.17: Clock – Enabled Global Buffer](image)

Now, for applications that only attempt to pause the clock for a few cycles on small areas of the design, the preferred method is to use the clock-enable pin of the FPGA register. The first example demonstrated in Figure 5.18 which illustrates an inefficient way of gating clock signals, while the second example in Figure 5.19 shows a modified version of the code that map efficiently into the clock-enable pin.

![Figure 5.18: Gated Clock – Not Preferable](image)

![Figure 5.19: Clock Enable – Efficient way of Gating a Clock Signal](image)
For global clock gating of design, one may use the clock-enable port of global clock buffers to stop the clock on entire clock domains. And for local level requirement, if clock gating of few registers of design is the required then following approach can be applied.

- Use the clock-enable port of registers to locally disable the clock.
- Consider replicating the clock-enable signal if it appears to be part of the paths that do not meet the timing requirements.

In this proposed design there is no scope for global clock gating as processor increments program counter on every clock and thus some portion of design is always active corresponding to the instruction fetched and executed. Thus for our processor design local level clock gating through clock enable pin is achieved. The basic idea over here is that for different types of instructions, different sets of registers are used, so when a particular instruction is getting decoded and executed the register corresponding to the other instructions toggle and get updated but are of no use but contributes heavily to consume the power. Thus whenever such situation occurs, it is better to do clock gating of registers to prevent them from toggling, in each pipeline stage, which are not in use by the current instruction. Following description explains that how the clock gating technique is implemented at all the stages of the pipelines.

Strategies to achieve the Clock Gating at different stages of pipeline

(1) Instruction Fetch

As program counter is computed and gets updated at every clock cycle; hence, there is no scope of Clock Gating.

(2) Instruction Decode and Operands Fetch:

In this stage instruction corresponding to program counter is fetched from ROM, since this is done at every clock cycle and the instruction fetched is combinatorially decoded. Hence there is no scope of Clock Gating.

(3) Execute Stage:
In this stage, the type of instruction is known; hence, accordingly registers corresponding to other instructions are Clock Gated through clock enable. There are three types of instruction getting executed in this stage, which are (a) RAM Instructions, (b) ALU Based Instructions and (c) Branch Instructions or NOP.

(a) **RAM Instructions:**

There are 4 RAM based instructions, so while these are loading or storing data to or from RAM, registers corresponding to ALU are Clock Gated. Instructions which require ALU operations generate regwrite signal in decode stage. This regwrite signal is used as clock enable signal for all registers corresponding to ALU based instructions. Thus when RAM instructions are getting executed regwrite will be zero and all the registers whose clock enable is fed by this regwrite signal will be disabled and considerable amount of power is saved. Registers which are Clock Gated during RAM instructions are as per the list given below:

- A (32 bit register)
- B (32 bit register)
- Regwrite2 (1 bit register)
- Zerowrite2 (1 bit register)
- Ra2 (4 bit register)

(b) **ALU Based Instructions:**

There are 28 instructions which use ALU during their execution, so all the registers which are used by RAM instructions can be Clock Gated. Thus clock enable of registers corresponding to RAM instructions is tied to ANDing of store and load signals generated in decode stage. Now if instructions are ALU based, load and store both will be zero and thus registers using ANDing of them as clock enable will remain disabled as shown in Figure 5.18 and Figure 5.19, and subsequently this arrangement turns into saving of power consumption. Registers Clock Gated during ALU based instructions are listed below:

- ram_datain (32 bit register)
- ram_addr1 (32 bit register)
- load2 (1 bit register)
store2 (1 bit register)

(c) **Branch Instructions OR NOP:**

Three branch instructions and an NOP instruction have no execution in execute stage so registers corresponding to RAM access and ALU all are Clock Gated. Regwrite, load and store signals all will be zero and thus above all registers will be clock gated. The Registers Clock Gated during Branch or NOP instructions are:

- A (32 bit register)
- B (32 bit register)
- Regwrite2 (1 bit register)
- Zerowrite2 (1 bit register)
- Ra2 (4 bit register)
- ram_datain (32 bit register)
- ram_addr1 (32 bit register)
- load2 (1 bit register)
- store2 (1 bit register)

(4) **Write back:**

In write back stage register bank of 16 registers of 32 bits are updated if an ALU based instructions was performed. Hence, in case of RAM, Branch or NOP instructions this update is not required and consumes power. So these registers are Clock Gated by connecting regwrite signal passed on from execute stage to clock enable pin of these registers. So if instruction was ALU based then only registers will be updated else they will remain disabled due to Clock Gating. Registers which are Clock Gated during Write Back Stage are:

- R0 (32 bit register)
- R1 (32 bit register)
- .
- .
- R15 (32 bit register)
After successful implementation of all the logic strategies for power optimization techniques called Novel RAM Addressing Scheme discussed in section 5.3.3 and Clock Gating described in section 5.3.4 at the processor architecture level, the CPU functionality has been tested and verified again to ensure its correctness. The power analysis of complete system has been done and the considerable amount of power reduction is achieved. It is resulted in reduction of 17mw (16%) of system power consumption. It is also noted that the Clock Gating technique is the major contributor in reducing the dynamic power consumption at the system level. The screen shot of the summary of power analysis is shown in Figure 5.20. Here the power analysis has been carried after implementation of two techniques together which are RAM Addressing Scheme and the Clock Gating along with earlier techniques.

Figure 5.20: Summary of Power Consumption Report for 4 – Stage Pipeline CPU (After Implementation of RAM Addressing Scheme and Clock Gating along with earlier Techniques)

The testing of Clock Gating has been done separately for all types of the instructions and are represented with the relevant waveforms as shown in following Figure 5.21 and Figure 5.22. Following waveforms describes the Clk_gating during for ALU related instructions.
Figure 5.21: Simulated Waveforms for Clk_gating Signal Verification for ALU Related Instructions
Figure 5.22: Simulated Waveforms for Clk_gating Signal Verification during RAM Access Instructions
It is shown in above Figure 5.21 that the registers such as data_ram, ram_addr, ram_data_in and load and store registers are clock gated during the execution of ALU related instructions and it also provides the constant hexadecimal “7FF” address which is fed to RAM during ALU based instructions, because of RAM Address technique implementation which provides ram_data_in to zero, the details of Clk gating during RAM Access instruction can be better explained through the waveforms shown in Figure 5.22.

As can be seen due to store instruction to RAM there is no change in a, b, ra, zerowrite, regwrite and register bank of 16 registers. When there is a Branch or NOP related instruction all the registers for both RAM Access and ALU are clock gated shown in Figure 5.22.

5.4 Verification of 4- Stages CPU After Implementation of Power Optimization Strategies

5.4.1 Performance Verification

All four power optimization strategies Memory Access Stage Removal, Resource Sharing Strategy, RAM Addressing Scheme and the Clock Gating have been implemented successfully at the hardware level and the whole system architecture has been tested and verified by executing many flavours of instructions and the performance of all of them has been checked for all the 4 – stages of the CPU. Following Figure 5.23 to Figure 5.26 demonstrates the performance of implementation after implementing all the power saving proposed techniques under different pipeline stages through the waveforms generated by using ModelSim SE 6.5. All these waveforms are self explanatory. Then after the power consumption comparison has also been made, in which the power consumption and estimated power for 5 – stage CPU has been discussed. Then the 5 – stage CPU is converted to 4 – stage CPU by implementing the Memory Access Stage Removal technique for which the power consumption is measured. Similarly, after implementation of each technique the power measurement is done and finally the power saving is achieved which is graphically represented through the comparison charts. The simulated waveforms for verification of 4 – stage CPU with Resource Sharing,Clock Gating and RAM Addressing Technique are demonstrated through following Figure 5.23 to Figure 5.26.
Figure 5.23: Verification of Instruction Fetch for 4–Stage CPU
Figure 5.24: Verification of Instruction Decode and Operand Fetch for 4-Stage CPU
Figure 5.25: Verification of Instruction Execute for 4 – Stage CPU
Figure 5.26: Verification of Write Back Operation of 4 – Stage CPU
5.4.2 Graphical Representation of Power Requirement

Following Figure 5.27 shows the graphical comparison of power requirements of the various modules of the systems for both the conventional (5 – Stage) and CPU with modified pipelining structure (4 – Stage), i.e. after implementation of Memory Access Stage Removal Technique. It is noted that the improvement in power consumption is achieved by 3mW (2.65%) in the system after the incorporation of this technique at the hardware design step.

![Power Summary for Conventional and Modified CPU](image)

**Figure 5.27: Graphical Representation of Estimated and Actual Power Consumption of 5 – Stage CPU**

Table 5.2 summarizes the results obtained from the power analysis carried on conventional 5 – Stage CPU and 4 – Stage CPU after implementation of Memory Access Stage Removal Technique. It represents the power consumed by the various modules of the system under consideration and the results are also compared with the estimated values.

**Table 5.2: Summary of Power Results**

<table>
<thead>
<tr>
<th>Category</th>
<th>Clock mW</th>
<th>Logic mW</th>
<th>Signals mW</th>
<th>BRAMs mW</th>
<th>MULTs mW</th>
<th>IOs mW</th>
<th>Leakage Power mW</th>
<th>Dynamic Power mW</th>
<th>Total Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Requirement</td>
<td>11</td>
<td>10</td>
<td>18</td>
<td>06</td>
<td>01</td>
<td>00</td>
<td>81</td>
<td>49</td>
<td>131</td>
</tr>
<tr>
<td>Conventional CPU (5-Stage)</td>
<td>04</td>
<td>03</td>
<td>18</td>
<td>05</td>
<td>00</td>
<td>00</td>
<td>83</td>
<td>30</td>
<td>113</td>
</tr>
<tr>
<td>Modified CPU (4–Stage)</td>
<td>03</td>
<td>03</td>
<td>17</td>
<td>05</td>
<td>00</td>
<td>00</td>
<td>83</td>
<td>28</td>
<td>110</td>
</tr>
</tbody>
</table>
The following Figure 5.27 shows the graphical view of the power consumption in different modules of the modified CPU after implementation of all the newly suggested power reduction techniques such as Memory Access Stage Removal, Resource Sharing, RAM Addressing Scheme and Clock Gating and finally the power dissipation comparison has made between the conventional 5 – Stage CPU and the modified 4 – Stage CPU modules. The power results indicating power consumption requirements are summarizes in Table 5.3.

Hence, the total power requirement of conventional 5 – stage pipeline CPU has been reduced to 89 mW from its original need of 113 mW i.e. overall 21% of improvement in the power consumption is achieved. The overall power comparison of the system has been presented in
the below Figure 5.29, which represents clearly that the power reduction is achieved successfully.

![Overall Power Consumption Comparison](image)

**Figure 5.29: Overall Power Consumption Comparisons**

The power consumption of these CPUs is also analyzed at different frequencies applied to the conventional CPU and the CPU with modified pipeline and also to the CPU after implementation of all the power saving proposed strategies, the related power figures are shown in Table 5.4.

<table>
<thead>
<tr>
<th>Frequency MHz</th>
<th>Power Consumption in 5 - Stage CPU (mW)</th>
<th>Power Consumption in 4 - Stage CPU (mW)</th>
<th>Power Consumption in 4 - Stage CPU after Implementation of all the Techniques (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>95</td>
<td>95</td>
<td>82</td>
</tr>
<tr>
<td>20</td>
<td>102</td>
<td>102</td>
<td>84</td>
</tr>
<tr>
<td>30</td>
<td>109</td>
<td>109</td>
<td>87</td>
</tr>
<tr>
<td>40</td>
<td>113</td>
<td>110</td>
<td>89</td>
</tr>
<tr>
<td>50</td>
<td>119</td>
<td>121</td>
<td>95</td>
</tr>
<tr>
<td>60</td>
<td>121</td>
<td>124</td>
<td>98</td>
</tr>
<tr>
<td>70</td>
<td>129</td>
<td>127</td>
<td>101</td>
</tr>
<tr>
<td>80</td>
<td>132</td>
<td>130</td>
<td>104</td>
</tr>
<tr>
<td>90</td>
<td>134</td>
<td>133</td>
<td>106</td>
</tr>
<tr>
<td>100</td>
<td>138</td>
<td>137</td>
<td>108</td>
</tr>
</tbody>
</table>

The plot for power consumption versus different clock frequencies for standard and the modified processors is shown in Figure 5.30.
It is noted from the Figure 5.30 that the power consumption increases as the clock frequencies increases for all the categories such as standard, modified CPU and the CPU after implementation of power saving techniques. But the power consumption of modified and optimized CPU implementation is always less than the standard CPU for all the clock frequencies.

5.5 Conclusion

In the beginning of this chapter the architecture of 5- stage standard CPU is discussed and its performance has been successfully verified through the waveform generation and analyzed for its power requirement using the Xilinx Xpower Analyzer tool.

In order to achieve the prime objective this 5 – stage pipeline structure of CPU is modified to have 4 – stage pipeline structure by implementing the memory access stage removal technique. Its performance is verified and power analysis has been done successfully and the reduction in dynamic power requirement is achieved without affecting the performance of the processor. Then all other newly suggested techniques which are Resource Sharing, RAM Addressing Scheme and the Clock Gating are implemented on the modified CPU and finally power analysis is carried on this CPU successfully.
It is concluded that the proposed techniques have impressively helped to reduce the dynamic power consumption of the implementation up to approximately 21% and that to without compromising the performance of the system.

Finally the various power comparisons are made using the data received from the analysis as an input and are represented graphically. Also the system behaviour is observed as per the expectation and the considerable amount of power reduction is achieved.

The implementation also analyzed for its power requirement at different clock frequencies and it is concluded that the power requirement for standard and the modified CPU is increased with the increase in the frequency but the modified CPU consumes always less power compared to the standard one.

Following chapter discusses the conclusions and the future scope of the work.