Chapter 5
RESULTS & DISCUSSION
CHAPTER-5
RESULTS & DISCUSSION

5.1 Introduction

For implementing the Adiabatic ASIP VHDL is used. This work deals with implementation issues involved with the design of Adiabatic ASIP architecture, including the use of entities and processes, the use of wait statements and signals, interfacing with different models, enforcing timing and dealing with asynchronous Adiabatic ASIP signals.

Design verification has been done at the pre-synthesis, post-synthesis and post-layout levels using Active HDL6.3 sp3. The Adiabatic ASIP was synthesized using Synplify Pro. Final layout was imported into Xilinx ISE 4.6 to perform design rule check.

For the implementation of adiabatic ASIP functional simulation was done using Active HDL6, logic synthesis for timing analysis was carried out using Synplify Pro, and Xilinx ISE for platform simulation. In addition, SPICE model is used in the design for modeling the adiabatic gates which in turn are used to design hierarchical blocks. The complete process for design simulation and verification required coordination and use of several CAD tools. Active HDL provides facility for use of integrated development environment(IDE) in order to include all the CAD tools in one interface for the whole design process. For logic synthesis, we are using Design Compiler of Synplify, because
it is still the most popular and almost standard logic synthesis tool used in industry. Also in the physical design step for simulation and verification waveform generation Active HDL 6.3 is used, which is also one of most popular physical design tools for rapid prototyping.

The Adiabatic ASIP has a low power internal architecture. The source code is written in VHDL. The VHDL model of Adiabatic ASIP is fully synthesizable with most synthesis tools and can be implemented in both ASICs and FPGAs. Some of the key features of Adiabatic ASIP are listed below:

- Dedicated ALU
- Hardware addition and subtraction
- Separate Instruction decoder and comparator
- Shifter Unit
- 8 bit I/O port
- On-chip 128 bytes Ram
- On chip Rom
- Debug Support

The first implementation of the Adiabatic ASIP architecture developed as a part of this work was an ALU that executed all instructions, except for memory operations, in a single cycle after being fetched. Later all other blocks were implemented and tested, finally these blocks were combined to generate the required Adiabatic ASIP.

Design debug and verification is one of the biggest challenges in the Adiabatic ASIP development. Simulation is a necessary step in the ASIC design flow. In order to verify the design has desired I/O behavior, simulation should be done at each step of the design
flow (system level, logic level, physical level). There are many types of simulation at different levels: Circuit Level, Switch level, Gate level, RT level, behavior level, HW/SW co-simulation, etc.

Maximum optimization at higher speed can be achieved at microarchitecture level by optimizing the task performing algorithms. Optimization at lower abstraction levels like circuit level or physical level are not as fast as higher abstraction levels but they yield maximum accuracy and closeness to design specifications. Until now, simulation is still the most dominant method for design verification. Here simulation technique is used for design verification. Adiabatic ASIP design flow approach is illustrated in Figure 5.1. The complete design of the Adiabatic ASIP blocks is done at the logical level through HDL coding and it is therefore independent of the circuit level technology. The block wise partitioning of design at logic level helps to identify and diagnose the errors. At this point simulation is used to verify that the HDL code of the Adiabatic ASIP blocks does not affect the functionality of the original design. A normal design flow approach is followed. First of all, functional simulation is carried out and then the verification switches to synthesis tool for timing verification. The switching from verification tool to synthesis tool continues back and forth until all the blocks of Adiabatic ASIP are tested and waveforms are carefully recorded. Since the logic level blocks of Adiabatic ASIP employs hierarchy, regularity, modularity and locality within the design therefore easy debugging and verification of logic blocks is possible. The entire implementation of our Adiabatic ASIP design consists of several major parts: Logic synthesis, Timing Analysis, etc. Figure 5.1 summarizes the process of designing, the process begins with a specification which describes in
a precise way what the device is expected do. This comes from the application (for instance if the device is to be designed to solve a specific problem), or may be market driven as in the case of devices built to suit consumer demand. The role of the designer is to extract from the specification a high level behavioral description of the circuit.

![Design Flow of Adiabatic ASIP](image)

**Figure 5.1 Design Flow of Adiabatic ASIP**

This level of abstraction organizes the system into several large tasks providing convenient divisions of the design effort. The designer also decides what parts of the design may be reused from an
assembled library of cores. Once the behavioral description of the circuit is clear, it must be translated to the RTL using a hardware description language such as VHDL. For some blocks automatic generation feature of CAD tools can be used, however automatic generation is not suitable for designing matured datapaths. Automatic generation can be successfully used for designing of memory arrays, random logics, control circuitry etc. still the handcrafted algorithms and circuits yields better results than their computer generated counterparts. The design complexity is increasing with advancements in VLSI design and with shrinking feature size. The increasing complexities can be dealt with design alterations at microarchitecture level for performance optimization. The design portioning is done at logic level (HDL descriptions) and as design is optimized using various algorithm based minimization techniques. The register transfer level gives an inside view of the design at block level instead of providing circuit level details. At RTL level the algorithms only define how the data moves within the datapath and its interaction with the controller. This is achieved by behavioral modelling of the partitioned design blocks.

Once the behavioral description of the Adiabatic ASIP blocks is completed and blocks are functionally tested and verified synthesis tool takes over and automatically generates the circuit level placement of components, wirings, interconnections etc. the synthesis tool (Synplify Pro) generates RTL blocks which are placed in logic order after timing analysis. Then interconnect and wiring equations are generated in the form of gate level netlist. The complete process of behavioral modelling, generation of blocks, functional testing, logical synthesis and finally generation of gate level netlist is very complex. It involves
switching back and forth from verification tool to synthesis tools and vice versa. There are every possible chances of errors creeping inside. For this at each step careful testing and debugging of the design has to be carried out. Sometimes synthesis output may be correct but the functionality of the block may not give desired results and requires switching to editor for making changes in code to achieve desired functionality. Modified design is again subjected to functional verification and again logical synthesis so a strong coordination between all the design verification tools is required. There exist several methods to deal with such design errors. Traditional verification methods along with simulation based methods are generally used for debugging the errors. Another approach for testing is through writing test-benches. Test-benches are efficient method of testing a design. Test-benches saves a lot of overhead of repeatedly testing a design. Test benches are programmed in HDL they apply input sequence to a logic model of the circuit to check functional operation of the block meets the required specifications. During simulation the higher abstraction level models can be translated to corresponding lower level of abstraction by shifting from one design domain to other design domain to obtain better insight of the circuit details. As the verification process moves from higher abstraction level to lower abstraction level the details of each block and henceforth complexity increases therefore simulation is more time consuming at lower level of abstractions. So it is always desirable to make optimizations right from the beginning that is from higher abstraction levels to all intermediate abstraction levels so as to optimize the simulation tool performance. The circuit optimization at each level of abstraction is desirable to identify and correct as many bugs as possible at higher level of abstraction thereby avoiding having
to spend more time and computational effort on design verification later on. Another reason for functional verification and testing at each abstraction level is required to be done because every time an error is found in code it has to be tested for functionality as well as for timing, routing by synthesis tool, this switching back and forth between editor, verification tool and synthesis tool increases computational overhead and latency. At some levels of abstraction errors can be corrected at the level itself and does not require switching between various tools. Therefore it has to be decided strategically that whether the design requires rechecking after error correction or not to save computational overhead and latency. Finally, design verification should ensure to the limits of:(1) accuracy of the simulation models and (2) Extensiveness of the test benches) the correlation between the specification and the fabrication instructions (layout).

From a logic-design point of view, the functionality of the ALU was trivial to achieve. Thus the focus was to minimize the power consumption. Following strategies were followed:

1. Minimize the area as much as possible via logic optimizations. Small area directly translates into low power design, thereby reducing power consumption.
2. Use adiabatic gates for modelling the ALU for adiabatic ASIP. Further optimize its performance by conducting experiments.
3. The overall ASIPs design’s performance will be set by its ALU and therefore, it was necessary to ensure that ALU is functioning smoothly in the entire design.

The functional verification results of all the basic 2-input logic gates (AND/NAND, OR/NOR, XOR/XNOR, Inverter) are given in figure 5.2 to 5.5.
Figure 5.2 Simulation Waveform of PFAL 2 input AND/NAND

Figure 5.3 Simulation Waveform of PFAL Buffer/Inverter

Figure 5.4 Simulation Waveform of PFAL 2 input OR/NOR
5.2 Adiabatic ALU

An 8-bit ALU can perform all the logic operations on two variables and a variety of arithmetic operations. ALU provides arithmetic operations like: add, subtract, compare, double, etc. Logic operations of two variables: XOR, compare, AND, NAND, OR, NOR, XNOR etc.

Figure 5.5 Simulation Waveform of PFAL 2 input XOR/XNOR

Figure 5.6 Logical Diagram of Adiabatic ALU
The multi-function ALU based on PFAL reversible logic gates. We reuse the output signals to reduce the cost of circuit design as much as possible. Figure 5.6 shows the logical diagram of adiabatic ALU.

5.3 Register Array

![Figure 5.6 Logical Diagram of Register Array](image)

Figure 5.7 Logical Diagram of Register Array

Figure 5.7 is logical diagram of register array that we have designed in active HDL 8.1. This register array has a ‘clk’ input signal, a select line ‘sel’, an enable signal ‘en’ and 16 bit input data line ‘data’. At the output it has a 16 bit output data line namely as ‘q’.

![Figure 5.8 Result Verification Waveform of Register array](image)
The above shown figure is the result verification waveform of register array. To verify the result, an input clock signal ‘clk’ is applied. To enable the register array an enable signal ‘en’ has to be given. Now when the select input is at ‘1’ and the data at the input lines is 0F0F, the output will be 0001.

### 5.4 Shift Unit

The figure 5.9 shows the logical block of a typical shift register designed for adiabatic ASIP. The logical block of shift unit has been designed using active HDL 8.1. The shift unit consist of entity architecture pair. In entity all the inputs namely an input bus ‘a’ and control signal select ‘sel’ along with output bus ‘y’ is shown.
Figure 5.10 Result Verification Waveform of Shift Unit

The figure 5.10 shows logical verification results shift unit of adiabatic ASIP. To verify the result, for the first 50 ns of time period when the select input is at ‘shiftpass’ and the data at the input lines is F00F, the output will be transparent i.e. F00F. Later at 100 ns of time period when the select input is at ‘shl’ and the data at the input lines is F00F, the output will be E01E.

5.5 Tri-State Register
The figure 5.11 shows the logical diagram of tri state register that we have designed adiabatic ASIP. This register consists of entity architecture pair. In entity all the inputs and outputs are declared. Namely a 16 bit input bus ‘a’, control signal ‘en’ and a clock signal ‘clk’. At the output end it has a 16 bit output bus ‘q’.

Figure 5.12 Result Verification Waveform of Tri Register
The figure 5.12 shows the result verification waveform of tri state register. To verify the result, a clock signal is applied at the ‘clk’ input of tri state register. Now for the first 50 ns of time period when the input clock signal is low and enable signal is at ‘0’ and the data at the input lines is undefined, the output will be the tristated or will be in high impedance state. Later at 100ns of time period when the input clock signal is low and enable signal is at ‘1’ and the data at the input lines is E001, the output will also become E001.

5.6 Bi Register

![Logical Diagram of Bi Register]

**Figure 5.13 Logical Diagram of Bi Register**
The figure 5.13 shows the logical diagram of bi register that designed in active HDL 8.1. This register consists of entity architecture pair. In entity all the inputs and outputs are declared. Namely an input bus has a 16 bit input signal ‘a’, and a clock signal ‘clk’. At the output it has a 16 bit output data bus as ‘q’.

Figure 5.14 Result Verification Waveform of Bi Register

The figure 5.14 shows the result verification waveform of bi register. To verify the result, a clock signal is applied at the ‘clk’ input of register. Now for the first 50 ns of time period when the input clock signal is low and the data at the input lines is at FF00, the output will be the FF00.
5.7 Comparator

The figure 5.15 shows the logical diagram of comparator designed in active HDL 8.1. This comparator has two 16-bit input buses namely ‘a’ and ‘b’, and a control input select ‘sel’. At the output it has an output data line ‘compout’.
Figure 5.16 Result Verification Waveform of Comparator

The figure 5.16 shows the result verification waveform of comparator. To verify the result, we have given 16 bit input at the two data lines ‘a’ and ‘b’. The input value at ‘a’ is 0F0F and at ‘b’ is also 0F0F. Since the comparator compares the two input data lines. Now when the select input is active the output of the comparator will be 1 and this can be verified from the above shown waveform.

5.8 Control Unit
Figure 5.17 Logical Diagram of Control Unit

The figure 5.17 shows the logical diagram of adiabatic ASIP control unit designed in active HDL 8.1. The signal description of control unit is given below in table 5.1.

Table 5.1 Signal description of Control Unit

<table>
<thead>
<tr>
<th>S.NO</th>
<th>SIGNAL NAME</th>
<th>SIGNAL TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CLK</td>
<td>INPUT</td>
<td>Clock Signal</td>
</tr>
<tr>
<td>2.</td>
<td>COMPOUT</td>
<td>INPUT</td>
<td>Signal from comparator</td>
</tr>
<tr>
<td>3.</td>
<td>INSTREGOUT</td>
<td>INPUT</td>
<td>Register selection line</td>
</tr>
<tr>
<td>4.</td>
<td>READY</td>
<td>INPUT</td>
<td>Ready Signal</td>
</tr>
<tr>
<td>5.</td>
<td>RESET</td>
<td>INPUT</td>
<td>Used to reset the system</td>
</tr>
<tr>
<td>6.</td>
<td>ADDREGSEL</td>
<td>OUTPUT</td>
<td>Address Register selection line</td>
</tr>
<tr>
<td>7.</td>
<td>ALUSEL</td>
<td>OUTPUT</td>
<td>ALU selection line</td>
</tr>
<tr>
<td>8.</td>
<td>COMPSEL</td>
<td>OUTPUT</td>
<td>Comparator Register selection line</td>
</tr>
<tr>
<td>9.</td>
<td>INTEREGSEL</td>
<td>OUTPUT</td>
<td>Register selection line</td>
</tr>
<tr>
<td>10.</td>
<td>OPREGSSEL</td>
<td>OUTPUT</td>
<td>Register selection line</td>
</tr>
<tr>
<td>11.</td>
<td>OUTREGRD</td>
<td>OUTPUT</td>
<td>Register Read Signal</td>
</tr>
<tr>
<td>12.</td>
<td>OUTREGWR</td>
<td>OUTPUT</td>
<td>Register write Signal</td>
</tr>
</tbody>
</table>
Figure 5.18 Result Verification Waveform of Control Unit

The figure 5.18 shows the result verification waveform of control unit. The control unit generates the various control signals to control the movement of data in ALU and other parts of the adiabatic ASIP. Control unit generates typical control signals like Clock Signal, Signal from comparator, Register selection line, Ready Signal used to reset the system, Address Register selection line, ALU selection line, Comparator Register selection line, Register selection line, Register selection line, Register Read Signal, Register write Signal, Program
Counter Read Signal, Program Counter write Signal, Read Signal, Register selection line, write Signal, Shift Register selection line, Output data line1 and Output data line2.

Figure 5.19 Logical Diagram of Adiabatic ASIP

Figure 5.20 Snap Shot of Logical Diagram of Adiabatic ASIP
5.9 Comparative Analysis

As mentioned earlier, the objective of this thesis was to design and implement an Adiabatic Microprocessor. To achieve this goal, it is required to have not only a working RTL model but also synthesize, place and route the design. This design posed various design problems. Therefore, for testing the functionality of the design after each design step verification, synthesis and routing was important. For this purpose, simulation of the design net list was done using Active HDL 8.1. Simulation results after the RTL design stage, synthesis, technology views were shown in Appendix.

A top down to bottom approach was used to simulate adiabatic ASIP and to test whether the design was properly synthesized and matches the specifications. Adiabatic logic circuits is the approach to minimize the power dissipation by recycling the energy stored in parasitic capacitance. With adiabatic circuits efforts are made that the energy stored in output capacitance is recovered to the power clock, unlike the conventional logic switching where the energy stored in output capacitor is dissipated as heat during discharge phase. This overcomes the inherent power dissipation due to contention current associated with skewed input signal transitions.

For improving power dissipation, the stray/parasitic capacitance plays an important role. Further the input bit pattern and transitions of input from logic ‘1’ to ‘0’ and ‘0’ to ‘1’ also causes either charging or discharging of load capacitor consuming energy. Power gating unused blocks of adiabatic ASIP can also result in considerable amount of power saving. The circuits operating in a conventional manner result in an energy dissipation that cannot be controlled by switching times. The input current waveforms of pull up or pull down in CMOS
technology shows unwanted current glitches on the controlled signal inputs which occurs when the NMOS or PMOS transistors turns on. Power consumption during this period is a significant factor contributing to the minimum energy performance of the adiabatic ASIP.

The circuits having greater logic depth this problem increases several folds. Conventional blocks and adiabatically controlled blocks are tied together to work in sync to improve the overall performance. This is achieved by strong communication between such blocks. However, in some cases when complexity increases too much such coordination among blocks may not be possible.

For true adiabatic operation it is necessary to avoid the crowbar condition of operation of MOS devices. Since the MOS devices have a fixed threshold voltage which should be applied to the MOS transistors to turn them ‘ON’ or withdrawn to turn them ‘OFF’ which causes a finite amount of current flowing through them for a brief moment. This contention current is undesirable for adiabatic operation.

The energy dissipation of this class of reversible circuit could be improved by decreasing the effective threshold voltage of the MOS devices. This would require a different MOS technology. There are factors that have to be considered when reducing the threshold voltage. Noise margin is decreased with the effect on reliability and increased static leakage currents could start to be the dominant cause of energy dissipation.

The performance of a range of reversible circuit elements operated in conventional and adiabatic logic switching has been studied. The major contributory factor to the limited adiabatic performance is identified.
Using transmission gates for implementing logic functions gives a better design approach to produce logically and physically reversible functions with the downside that implementations using transmission gates are non-restoring in nature. However in certain design blocks where signal driving capabilities are not significant issue transmission gate based implementations are good alternatives.

The functional verification waveforms of PFAL based adiabatic arithmetic units described above are presented and discussed in this section. For comparative analysis, the data of CMOS ASIC and CMOS ASIP for the same technology has been used. For the adiabatic ASIP a power clock is used and it is assumed for simulation purpose that efficiency of power clock is 90%. The input test pattern is a long sequence of random values that gives a figure of the average power consumption.

Table 5.2 Power Comparison of Adiabatic ASIP with CMOS ASIC and CMOS ASIP

<table>
<thead>
<tr>
<th>Power [mW] @30MHz</th>
<th>Adiabatic ASIP</th>
<th>CMOS ASIC</th>
<th>CMOS ASIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power [mW] @20MHz</td>
<td>2.22</td>
<td>6.28</td>
<td>6.36</td>
</tr>
<tr>
<td>Power [mW] @10MHz</td>
<td>2</td>
<td>5.86</td>
<td>6.01</td>
</tr>
</tbody>
</table>

Figure 5.21 Power Comparison of Adiabatic ASIP with CMOS ASIC and CMOS ASIP
A. Energy and Power Characterization:
The table 5.2 gives Power Comparison of Adiabatic ASIP with CMOS ASIC and CMOS ASIP. The energy consumption per operation and the adiabatic gain of the PFAL arithmetic units working at 10, 20, 30 MHz are reported in figure 5.21. The consumption of the equivalent CMOS ASIC and ASIP are also reported for comparison. Since we are dealing with a semi-custom design built of fixed cells, the circuits are not optimized for the particular point of operation, such as modifying transistor dimensions or scaling the supply voltage. The results show that semicustom adiabatic ASIP unit is more energy efficient over the entire frequency range.

B. Comparisons:
The results are also compared for the different types of instructions with other CMOS based ASIC and ASIP units published in the literature [11]–[19].

![Figure 5.22 Power Consumption Comparison for Different Scenarios @ 10 MHz](image-url)

**Figure 5.22 Power Consumption Comparison for Different Scenarios @ 10 MHz**
Even if technology, power supply, and simulation methodologies widely vary, it is common that every author compares its solution with an equivalent CMOS circuit. Therefore, we can use the adiabatic gain as a good parameter to fairly compare the various solutions. Fig. 5.22 and 5.23 show, the power consumption by different types of instruction for two frequency cases: 10 and 20 MHz.

In any case, Fig. 5.22 and 5.23 show that the power consumed by the adiabatic ASIP is the lowest even if they are designed with a semicustom technique. It is worth noting that the significant energy dissipation reduction is achieved with a design methodology consisting of conventional design steps and tools the designer is used to and confident with.