Chapter 7

Conclusion and Future Work

Efficient designs are developed for high performance decimal floating point MAC unit as part of this research. The modified reversible BCD adder implementations presented are highly optimized in terms of number of reversible gates and garbage outputs. The new exhaustive branching algorithm obtains reduced hardware and/or delay for logic implementation using multiplexers for a VLSI design. The simulation results and the performance analysis show that the designs presented in this research are suitable for improving the performance of architectures for decimal computations. Hence these design techniques and circuits are dependable alternatives that could be used for high performance decimal processors. Suggestions for further work in this field are also presented.
7.1 Conclusion

Efficient design methods and architectures are developed for a high speed Decimal Floating Point (DFP) Multiply Accumulate (MAC) unit as part of this research. The decimal MAC unit has a multiplier fused with an adder module. The multiplier has to be an efficient, high speed multiplier for the MAC unit to achieve high performance. This research presents two novel techniques for iterative DFP multiplication. The first approach has a Decimal Fixed Point (DFxP) multiplier using a novel Double Digit Decimal Multiplication (DDDM) technique that performs two digit multiplications simultaneously. The speed to complete an n-digit × n-digit multiplication is almost doubled compared to a single digit design at an expense of 50% increase in area for this design. The design was validated using lengths of 7-digit, 16-digit, and 34-digit multipliers that are required correspondingly for decimal32, decimal64, and decimal128 formats of DFP multiplications. In addition, 34-digit multiplier is designed as a partitioned combination of 17-digit multipliers. For the partitioned implementation, the speed is increased compared to the iterative DDDM implementation.

The second iterative approach does DFxP multiplication using a novel RPS algorithm. In this approach, partial products generated using BCD digit multipliers are accumulated from the least significand end in a column manner. This design leads to a more regular implementation, and does not require special registers for storing multiples of multiplicand. The simulation results show that the RPS design gives a reduction in delay compared to the single digit implementation for 7-digit × 7-digit DFxP multiplier. This
iterative approach is suitable for high speed DFP multiplication since rounding can be initiated during the DFxP process.

A novel design for BCD digit multiplication that reduces the critical path delay and area is also presented in this research. The design for a BCD digit decimal multiplier is extended to a Hex/Decimal digit multiplier that gives either a decimal output or a hex output depending on the requirement. The comparison shows that the new design of Hex/Decimal multiplier has a reduction in delay compared to the existing design.

The iterative DFP multipliers using floating point extensions of the newly designed iterative DFxP multipliers use DDDM technique and RPS algorithm. A delay reduction is achieved using RPS algorithm because of the initiation of the rounding process during the DFxP multiplication. This parallelism decreases the worst case time period. The design using DDDM is more regular and occupies lesser area, but has more delay compared to the design using RPS algorithm.

The research also presents parallel DFP multiplier using modified parallel DFxP multiplier for significand digit multiplication. Parallel designs are adopted when latency and throughput are considered more important than area. The parallel DFP multiplier includes the floating point extensions of the parallel DFxP multiplier design. In this modified parallel DFxP multiplier design partial product accumulation is done using both row and column accumulations. The column accumulation approach gives a decrease of area and delay over the row accumulation.

Floating point MAC unit is designed for fused multiply-add operation with a single final rounding after add operation. The fused multiply-add unit uses parallel or iterative multipliers and a floating point adder unit. The DFP
adder is designed using ripple carry BCD adders, Kogge-Stone adders and 'reduced delay BCD adders'. Comparison of DFP adders shows that the 'reduced delay adder' achieves the highest speed.

In recent years, reversible logic has emerged as one of the most important approaches for power optimization. So, reversible logic is in demand in high-speed power aware circuits. The modified reversible BCD adder designs presented in this research are highly optimized in terms of number of reversible gates and garbage outputs. The comparison with existing designs in literature shows that the modified designs use least number of gates, produce least number of garbage outputs, and give least levels of delay. Speed of reversible design for carry select and hybrid BCD adders are compared with a conventional BCD adder. The results reveal that the hybrid BCD adder attains speed up over other two designs, for any input length in reversible implementation. This research also presents a reversible fault tolerant design using Fredkin gates (FRG) for conventional BCD adders, adders for QAD, and carry select BCD adders for multi-digit BCD addition. Toffoli Gate (TG) designs for multi-digit addition are also presented. The implementations using Toffoli gates give superior results in terms of quantum cost, garbage count and gate count, compared to Fredkin gate implementations.

A new reversible 4-bit Binary to BCD converter circuit that requires only one gate without garbage output is designed using a novel reversible 4×4 RPS gate as part of this research. A reversible BCD adder is also designed using this RPS gate that replaces the '6-correction circuit' and the 'final 4-bit binary adder'. In addition, a new partially reversible RPS gate that satisfies the reversibility criteria for BCD inputs is designed. This gate can further reduce the number of logical computations involved in BCD arithmetic reversible
circuits. The reversible implementations of BCD adder using combination of HNG-RPS (fully and partially) gates and using HNG gates are also presented in this research. A reduction in logical complexity is achieved by HNG-RPS design compared to the existing reversible BCD adder designs. It is also seen that the combination of HNG-RPS gates makes the BCD adder design more compact and reduces the number of garbage to a near optimal value.

Low power circuits designed in reversible logic for (7, 4) Hamming code generation and error detection circuits are presented in this research. The logic gates of a classical logic implementation of Hamming code generation and error detection circuits are replaced with reversible equivalents. Among these designs, the one using new $4 \times 4$ HCG is highly optimized in terms of number of reversible gates and/or garbage outputs. The use of new parity preserving HCG (PPHCG) gate provides a way of incorporating fault tolerance into reversible circuits with modest hardware overhead. Parity preservation by itself proves useful for ensuring the robustness of reversible logic circuits in their various application domains.

The new exhaustive branching algorithm suggested in this research obtains reduced hardware and/or delay for logic functions using multiplexers for a VLSI implementation. Exact functions are realized since the number of variables is also given as an input.

The simulation results and the performance analysis show that the design approaches and the architectures presented in this research are suitable for improving the performance of decimal computations. Hence these design techniques dependable alternatives that could be used for high performance decimal processors.
7.2 Suggestions for Future Work

Suggestions for further investigations in the field of architectures for decimal computations in continuation with the present work are listed below.

- The designs presented use BCD encoding for decimal representation which is the simplest and most popular code for decimal data. Alternatively, other decimal representations such as BCD Excess-3 (BCD XS3) representation may be used which may possibly allow more efficient decimal addition/subtraction.
- The use of specialized encodings such as 4221, 5211, 5421 codes that may improve the speed of computation in a decimal ALU can also be explored. If these specialized codes are suitable only for certain computations in the decimal ALU then the use of efficient intermediate encodings for processing decimal data may also be researched.
- Signed digit encodings that may reduce the number of partial product accumulations may be investigated. This may increase the speed of computations.
- The development of decimal/binary processor using these new designs may be explored.
- Future research may focus on incorporating more pipelining to improve the speed of multipliers and fused multiply-add computations.
- Custom layout design that takes advantage of high-speed and low-power circuit techniques may be developed.
- The reversible circuits presented in this research may be used to develop a decimal ALU for a reversible CPU.
- Further investigation into determining reversible implementations using logic synthesis methods [A. Agrawal and N. K. Jha, 2004],
[Dmitri Maslov, 2003], [P. Gupta, A. Agrawal, N.K. Jha, 2006], [G. Yang et al., 2006] may be studied.

- Design modifications to reduce the total worst case delay may be investigated by varying the size of carry look-ahead blocks in a hybrid reversible adder.

- The investigation may focus on architectures for multipliers, comparators, etc using the fully and partially reversible RPS gates.

- Additionally, it is noted that there is a lack of simulation tools that support reversible gates, and this is most definitely an area worthy of attention.

- The exhaustive branching algorithm may be extended for the synthesis of incompletely specified functions. Research may be done for an alternative design of logic functions using different size multiplexers.