MB-LITE: A robust, light-weight soft-core implementation of the MicroBlaze architecture

Tamar Kranenburg
Delft University of Technology
EEMCS, Circuits and Systems group
Mekelweg 4, 2628 CD Delft, The Netherlands
Email: info@takar.nl

Rene van Leuken
Delft University of Technology
EEMCS, Circuits and Systems group
Mekelweg 4, 2628 CD Delft, The Netherlands
Email: t.g.r.m.vanleuken@tudelft.nl

***Abstract*—Due to the ever increasing number of microprocessors which can be integrated in very large systems on chip the need for robust, easily modifiable microprocessors has emerged. Within this paper a light-weight cycle compatible implementation of the MicroBlaze architecture called MB-LITE is presented in an attempt to fill the gap in quality between commercial and open source processors. Experimental results showed that MB-LITE obtains very high performance compared with other open source processors while using very few hardware resources. The microprocessor can be easily extended with existing IP thanks to an easily configurable data memory bus and a wishbone bus adapter. All components are modular to optimize design reuse and are developed using a two-process design methodology for improved performance, simulation and synthesis speeds. All components have been thoroughly tested and verified on a FPGA. Currently an architecture with four MB-LITE cores in a NoC architecture is in development which will be implemented in 90nm process technology.**

I. Introduction

The increase in capacity of Field Programmable Gate Arrays (FPGAs) made it possible to include many microprocessors within a System On Chip (SOC). Concepts in the field of Parallel Computing and Network on Chip architectures can be simulated using FPGA technology. Both commercial as well as open source processors have been used to this extend [1], [2], [3], [4]. However, a commercial processor (e.g. MicroBlaze from Xilinx or NIOS2 from Altera) poses stringent restrictions on portability since it is expensive or even impossible to fabricate it on silicon - or even on other devices. Furthermore, including very specific components like for example a custom data bus is a tedious task.

Open source processors do not have these limitations, but aspects like design quality, support and documentation can become a bottleneck. In [5] it was concluded that most open source designs are hard to modify due to their complexity, the lack of a unified design methodology or the absence of clear documentation. Furthermore, most of these microprocessors are not portable and will only work on specific FPGAs.

In this paper the design and implementation of MB-LITE is presented in an attempt to provide an easily modifiable high quality open source processor. MB-LITE is a light-weight implementation of the MicroBlaze architecture [6]. The microprocessor is cycle and architectural compatible with the MicroBlaze specification. A wishbone bus and easily configurable data bus are provided. All components are modular, making it very easy to modify the memory topology. A two-process design methodology is used to optimize readability, performance and synthesis.

The remainder of this paper is organized as follows. In Section II several existing processors are discussed. Section III will focus on the design of MB-LITE while in Section IV the performance with respect to several other cores is discussed. Finally several conclusions will be presented in Section V.

II. Background

A. *Evaluation of open source processors*

In [5] several open source processors were evaluated. Aspects like design quality, performance, available documentation and the set of available features were taken into account. For the development of prototypes it is important that the design can be quickly modified so the design needs to be simple and flexible. The processor needs to be robust and a tool chain with standard libraries need to be available. A single-threaded, single-issue RISC architecture meets this goal best. Four processors were selected for further investigation (OpenFire, OpenRISC, LEON3 and aeMB). Additionally a comparison was made with MicroBlaze.

LEON3 is an implementation of the relatively complex UltraSPARC architecture and has many features. A drawback is that it has a bus (AMBA) which is rather hard to adapt. Furthermore, the architecture is quite complex so it will in general be hard to make modifications. Both OpenFire and aeMB implement the MicroBlaze architecture but do not have many features. Both processors would need improvements in terms of portability and available features. However, the documentation of both processors is shallow and no design methodology was applied. Improving these processors implies that thorough insight in the design need to be gained, which is tedious and time consuming. The resource utilization of all processors was measured and their relative performance was compared using the execution time of the Dhrystone benchmark. All optional components and features were disabled. It turned out that existing open source processors have low to moderate performance, while resource utilization is

Listing 1. Example of the two process design methodology. Signals r, rin and variable v are of the same type, hence they can be easily assigned to each other eliminating the need for long assignment lists.

```
fetch_o <= r;

PROCESS(fetch_i, r, rst_i)
  VARIABLE v : fetch_out_type;
BEGIN
  IF rst_i = '1' THEN
    v.program_counter := (OTHERS => '0');
  ELSIF fetch_i.hazard = '1' THEN
    v.program_counter := r.program_counter;
  ELSIF fetch_i.branch = '1' THEN
    v.program_counter := fetch_i.branch_target;
  ELSE
    v.program_counter :=
      increment(r.program_counter);
  END IF;
  rin <= v;
END PROCESS;

PROCESS(clk_i)
BEGIN
  IF rising_edge(clk_i) THEN
    IF ena_i = '1' THEN
      r <= rin;
    END IF;
  END IF;
END PROCESS;
```

medium to high. Comparing these results with commercial implementations justifies the development of MB-LITE.

B. *Structured* VHDL *design*

Traditional VHDL design approaches contain many small processes and do not follow a unified naming convention. As a result these designs are hard to understand and tedious to maintain [7]. No benefit is taken from the increased level of abstraction offered by behavioral VHDL models.

A good solution to the bottlenecks of such ad-hoc designs is also given in [7]. Combinational and sequential elements are explicitly separated to clarify time dependencies between processes. The algorithm is completely determined by the combinational process. Component results are stored in the sequential process. Due to this separation a synthesizer can take more advantage of its optimizing capabilities. As a result the models become easier to understand and become less error-prone. Using a high abstraction level is recommended to keep the code size to a minimum while gaining a much better structure. As a consequence these designs will be easier to understand, debug and maintain. A fragment of a MB-LITE component modeled using this methodology is shown in Listing 1.

C. *The MicroBlaze architecture*

MicroBlaze is a RISC architecture and almost identical to the MIPS architecture of early 1980s. It is a Harvard architecture with 32-bit instruction and data words. The architecture is designed to be implemented with five pipeline stages. Most instructions - except for branches - have a latency of one cycle. The architecture provides a single interrupt. A few notable differences between MicroBlaze and MIPS will be discussed in this section.

A conditional branch in the MIPS architecture consists of two subsequent instructions. First, an arithmetic operation is executed and second the branch is performed based on the result of the previous instruction. Therefore just a single adder and a comparator is necessary to execute a branch. MicroBlaze does both the evaluation of a condition as well as the computation of the target address within a single instruction, hence more hardware is required.

The MicroBlaze architecture need the values of three registers at the same time, since the definition of store word is MEM[R_a+R_b] = R_d. All three register values need to be forwarded which requires additional logic compared with the MIPS architecture.

III. Design

A. *Organization*

The organization of MB-LITE is based on the MIPS processor described in [8] because this is a well-known implementation and the architecture is straight-forward. It has the same pipeline stages Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory (MEM) and Write-back (WB). Several modifications to the MIPS organization are applied in order to obtain a MicroBlaze compatible implementation.

MB-LITE implements distributed control in order to eliminate the need for a centralized and complex pipeline controller. All dependencies like stalls, hazards and forwards are solved locally. It is therefore relatively easy to understand the design, since the dependencies have been made clear and simple to understand. For example, the signal alu_src_a for example is evaluated and used in the execution stage and depends on the decoded operation and the register values being forwarded by other stages. Using this approach, the logic necessary for selecting the correct values of the ALU operands is straightforward. Almost all forwarding logic is collected in the execution stage in order to obtain a structured, strictly synchronous design.

The ID stage decodes the instruction into clear, well defined signals. These control signals travel along with the instruction. The execute stage set the ALU operands and the operation which needs to be executed. Besides all basic functionality like shifting, adding and logic operations the ALU can be extended with a multiplier and barrel shifter.

Several techniques are applied to solve hazards. For data hazards forwarding is applied to reduce the number of stalls to a minimum. The structural hazard which occur when the same register is read and written concurrently is also solved using operand forwarding. When the result of a load instruction is immediately used, these techniques can not be applied and a stall will be inserted in the pipeline. Finally, control hazards are solved using a pipeline flush.

B. *Data memory bus*

To simplify connecting components like memories, co-processors, bridges and adapters an easily modifiable address decoder was designed. The memory map and the number of outputs can be configured using VHDL generics. The bus is

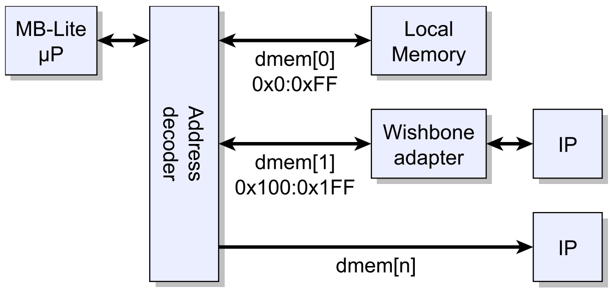

Fig. 1. Example of a memory topology build using the address decoder. Fast local memory is connected to the lower part of the memory map, while multi cycle transactions can be connected using the wishbone bus adapter.

responsible for decoding the selected address and steering the control, address and data signals to the appropriate output ports. The decoder can be connected directly to the MB-LITE data bus and does not introduce additional delay.

Since many peripherals for the wishbone bus are available an adapter is developed to convert the MB-LITE data memory side of the processor to a wishbone master interface. The adapter can be connected directly to the MB-LITE core or to the decoder. The wishbone bus introduces a delay of one cycle in order to avoid asynchronous control paths. An example of a memory topology including the address decoder and bus adapter is shown in Figure 1. The wishbone bus is modeled in accordance with revision 3B of the wishbone specification. During a wishbone cycle the execution of the processor is interrupted until the transaction is finished.

The data memory interface consists of basic control signals and can be connected to any type of synchronous memory. For halfword and byte data transfers a memory with four independent WRITE ENABLE inputs is needed. Since such a component is not available on every platform a memory consisting of four independent memory components is included as well. The data interface of MB-LITE has a 4-bit select signal which can be used for reading or writing specific bytes.

IV. Results

A. Verification

The behavioral model of MB-LITE has been extensively verified for proper functionality. A character device was designed and connected to the data interface of MB-LITE in order to receive feedback. This interface, which uses the TXTIO library, is capable of printing characters to the console of a simulator.

To verify the design a large test bench have been carefully assembled and obtains full statement coverage. All statements in the behavioral VHDL model are executed at least once during this simulation, which makes the execution of individual instructions very reliable. During this simulation a wide range of programs are executed and the output is compared with expected results. Several standard libraries like STDIO.H, MATH.H and FLOAT.H are included in this test bench as well as often used functions such as STRCPY and MALLOC. The test bench have been executed with and without the optional multiplier and barrel shifter.

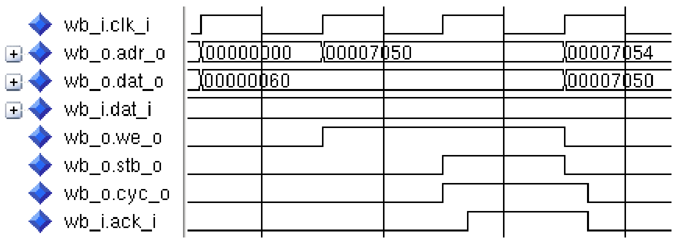

Fig. 2. Plot of a wishbone write cycle.

The data bus components have been tested using the same procedure. A memory topology was set up using the address decoder and the wishbone bus. The lower part of the memory was connected to block RAM while the upper part of the memory connected through the wishbone bus adapter to a wishbone compatible IO device. During the tests of the wishbone bus the acknowledge delay of the bus was altered to see if all rules and exceptions of the wishbone bus specification have been met. An example of a wishbone write cycle is shown in Figure 2.

A VHDL netlist was generated from the behavioral model using ISE 10.1. The functionality of the synthesized design was successfully verified by comparing the simulation of the behavioral model with the simulation of the synthesized model with time annotation.

B. Performance

The performance of the MB-LITE processor has been compared to several other designs. All designs were synthesized using a Virtex 5 development board (XC5VLX110-3FF1760) using Xilinx XST 10.1.03. To obtain a valid comparison all processors have been configured with as few as possible features. Multipliers, barrel shifters, bus controllers and interrupts amongst others have been disabled in all designs. As a result, we measure the performance of the bare core.

Performance was estimated in terms of clock frequency and execution time of the Dhrystone benchmark. To this extend the Dhrystone benchmark has been ported to MicroBlaze by replacing functions with equivalent MicroBlaze routines. The results of these measurements are shown in Figure 3. AeMB seems to obtain a very high clock frequency, but in the end it takes a lot of time to execute the Dhrystone benchmark. This is due to the fact that AeMB is designed for interchanged execution of two-threads. Due to this time multiplexing many dependencies can be removed and the clock frequency can be much higher compared with general designs. Since we are primarily interested in the execution of a single thread we consider this as a feature rather than a possible performance advantage.

Although the clock frequency of OpenRISC does not deviate that much from the other designs, the real execution time is very high. Besides, it uses a considerable amount of hardware resources. It was found that the both the architecture as well as the tool chain are not very efficient. The performance of LEON3 is quite disappointing. Apparently the tool chain can

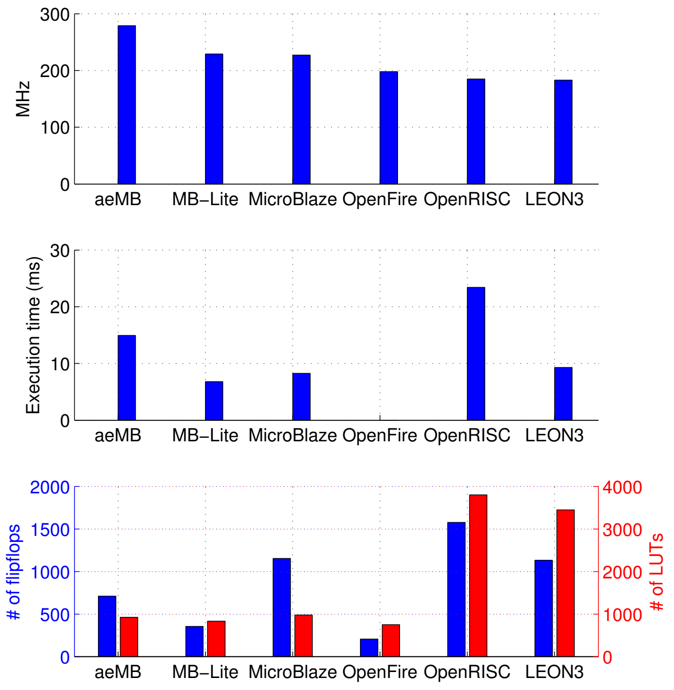

Fig. 3. Clock frequency, execution time of 1000 iterations of the Dhrystone benchmark and resource utilization of several open source processors. Simulation results of OpenFire could not be obtained.

not take much benefit of the SPARC register windows. The execution time of MB-LITE is considerably lower than all other designs, including the commercial implementation of Xilinx. Furthermore, the resource utilization is very low.

C. *Realizations*

An implementation of MB-LITE in a Xilinx Virtex5 consisting of a configuration of the MB-LITE core, memory, multiplier, barrel shifter and an 16550 compatible UART results in a clock speed of 65 MHz and a resource usage of 1450 LUT's, 3 multiplier blocks and BRAM36 blocks. Implementations with other peripherals produce about the same numbers. The software which was used to test these implementations was generated by the standard MicroBlaze tool chain and used without modification. Currently a configuration of four MB-LITE cores in a NoC architecture is in the design stage. The goal is to realize this design in 90nm CMOS process technology.

V. Conclusion

In this paper the design and implementation of an easily modifiable open source processor was presented [9]. MB-LITE is a light-weight implementation of the MicroBlaze architecture and is designed to obtain high performance using few logic elements. This has been achieved by applying a synchronous two-process design methodology in which combinational and sequential logic is strictly separated. Comparisons with other processors show that MB-LITE and the implementation of Xilinx have equivalent performance, while using far less resources.

Conformance with the MicroBlaze architectural specification was thoroughly verified using both behavioral as well as netlist simulations with annotated time information. The design is cycle as well as architecturally compatible with MicroBlaze. MB-LITE might thus be used to replace MicroBlaze in most designs in order to make them better portable.

The two-process design methodology is not only used to improve design speed, but also to increase readability, facilitate code maintenance, reduce code-size and improve simulation and synthetization speeds. Using modularity as basic rule, a highly configurable multiplexed bus and wishbone bus adapter can be easily attached and modified.

All components are inferred instead of explicitly instantiated. Therefore this design can be much easier ported to other technologies. The separation between standard and specific components and the use of strict synchronous components facilitates the implementation of MB-LITE in an IC fabrication process without having to change the structure of the design.

The maximum clock frequency of MB-LITE is comparable with MicroBlaze, while the execution time of the Dhrystone benchmark is much lower than all evaluated processors. Besides, the resource requirements of this processor are far less than many other designs, while the core provides at least the same functionality. The core can be configured to use a multiplier or barrel shifter and implements a single interrupt.

The small size of the MB-LITE processor, the modularity of the bus as well as the application of the two-process design methodology makes this processor very well suited for research and development of Very Large Scale Integrated Systems On Chips and On-Chip Networks. Future research will focus on embedding the MB-LITE in a reconfigurable fabric as well as implementing this processor in a UMC 90 nm process technology.

References

[1] D. Sheldon, R. Kumar, F. Vahid, D. Tullsen, and R. Lysecky, "Conjoining soft-core FPGA processors," *Computer-Aided Design, International Conference on*, pp. 694–701, 2006.

[2] F. Plavec, B. Fort, Z. G. Vranesic, and S. D. Brown, "Experiences with soft-core processor design," in *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2005, p. 167.2.

[3] R. Holsmark, A. Johansson, and S. Kumar, "On connecting cores to packet switched on-chip networks: A case study with MicroBlaze processor cores," 7th IEEE Workshop DDECS 04, april 2004.

[4] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[5] T. Kranenburg, "Reference design of a portable and customizable microprocessor for rapid system prototyping," Master's thesis, Delft University of Technology, 2009.

[6] *MicroBlaze Processor Reference Guide*, Xilinx, January 2008.

[7] J. Gaisler. Fault-tolerant microprocessors for space applications. [Online]. Available: http://www.gaisler.com/doc/vhdl2proc.pdf

[8] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. Morgan Kaufmann Publishers, 2003.

[9] T. Kranenburg. Mb-lite project. [Online]. Available: http://www.opencores.org/project/mblite