ABSTRACT

This paper presents the design and FPGA implementation of a 32-bit configurable micro controller. The micro controller contains a 32-bit processor based on RISC-V Instruction Set Architecture, Cache memories, interrupt support, multiplexed buses and a Debug Unit. The processor supports all integer arithmetic. Cache memories have various sizes up to 16kB. Prioritized stacked interrupt control is present. The design achieves a peak frequency of 80MHz and utilizes 14,462 Look-Up-Tables (LUTs) on Virtex-7 FPGA.

Keywords: RISC-V, Processor, Cache, Interrupt

I INTRODUCTION

Micro controllers are found in a plethora of applications from toys to industrial control systems. Most chips have proprietary Instruction Set Architectures (ISA) which makes customization difficult and expensive. Most micro controllers are suited to a particular set of applications. A system based on an open ISA and having customizability is beneficial for fast and variety of development.

The implementation is a framework to develop a full fledged System-on-Chip. Based on an open ISA[1] and having support for extensions provides greater flexibility. The is implemented to be able to function standalone with/without external memory and run reasonably complex applications.

II RELATED WORKS

System-on-Chip development in the field of RISC-V started with the release of “Rocket Chip”[2]. It is tethered to an ARM core and requires intervention from the core to emulate DRAM and peripherals.

In [3], the designers have released a soft core in chisel, called “ZSCALE”, following the RISC-V RV32IM extension. The “VSCALE”[3] is the Verilog version of “ZSCALE”.

“mRISC-V”, a RISC-V based micro controller, is described in [4]. [3] and [4] do not have caches or atomic extensions.

III HARDWARE IMPLEMENTATION

The implementation with independent blocks is shown in Figure 1. It contains a 32-bit processor following RV32IMA extensions of RISC-V, Instruction and Data Cache, hardware timers, WISHBONE bus interconnect, configurable peripherals, pre-emptive interrupt handling and a debug unit.

3.1 Processor

The processor is a 32-bit 5-stage pipeline. The stages are fetch, decode, execute, memory and write-back. It is shown in Figure 2.

It follows the RISC-V RV32IMA extensions. This implies that the hardware supports integer arithmetic/logical operations, multiply/divide execution and Atomic Read-Modify-Write memory operations. Read/write to memory is also allowed with byte addressability.

Processor also has interface to interrupt controller which performs handshakeing and saving/restoring state. System counters are embedded in the processor. There are 3 counters: real time, clock counter and instruction counter.

3.2 Memories

The implementation has a Modified-Harvard architecture with separate instruction/data caches and a common main memory. Each cache memory’s configuration is 16kB, 2-way.
size of cache is 32-byte. Transfers between cache and main memory are burst-of-8 words.

The Data cache can be configured in size and enabled/disabled in software.

### 3.3 Bus Transport System

The interconnect topology is a star topology with multiplexed master and slave buses. It follows the WISHBONE B.3 standard. A bus controller supervises all transactions. Burst and classic transfers are allowed with burst transfers for memory and classic transfers for memory mapped I/O and peripherals.

### 3.4 Interrupt Control

Interrupt is a mechanism by which a peripheral can suspend the normal execution of processor and get itself serviced. The interrupt request generated by the peripheral causes the CPU to stop what it is doing and jump to a separate piece of code called an ISR (Interrupt Service Routine). When the ISR is finished, it returns to the code that was running prior to the interrupt. The interrupt controller receives interrupts from peripherals and sorts them according to priority. It then updates its internal registers and signals the CPU by sending an “IRQ request”. The CPU stores the current state of the program onto the stack and then sends an “ACK” signal acknowledging the interrupt controller. Following this, CPU jumps to a common piece of code where it reads the Status Register and jumps to the specific ISR based on the interrupting device and services it. The encounter of ERET instruction marks the end of the ISR, following which the state of the program prior to the interrupt is restored from the stack and normal program execution continues.

### 3.5 Debug Unit

Modern software contains bugs, and to help find these bugs it is critical to have good debugging tools. The processor debug unit assists in debugging software running on the processor. It provides visibility into what is going on in the processor core. The communication with the host is done with a two-wire cable following the UART (Universal Asynchronous Receiver/Transmitter). To allow for debugging, an On-Chip Debug Unit (OCD) with Debug Support Unit (DSU) and Debug Handle Unit (DHU) is designed. The DSU has registers which can be used for controlling the program flow like halting the CPU, setting a breakpoint, single stepping. The DHU controls instruction flow across pipeline.

### 3.6 Error correction

Memory data corruption is often fatal to the operation a system. In a processor-based system, memory errors result in incorrect values in either instruction or data streams. Errors in data streams may cause the program flow to derail. An error-correcting code uses multiple parity check bits that are stored with the data word in memory. The error correcting code used here is SEC-DED. The parity bits are computed and stored in memory along with data. On memory read, the parity bits are recomputed. Mismatch indicates data stored in memory is corrupted.

### IV Results and Analysis

Design and implementation is done in Xilinx Vivado 2015.4 with Virtex-7 FPGA as target. Table IV shows the resource utilization for the blocks obtained after Place and Route. The Top module contains processor with RMW enabled, D-Cache, I-Cache, Interrupt controller, clock generator, Bus controller with 2 masters,4 slaves and 64kB main memory.

Table IV shows the comparison of our processor with VScale processor from UC, Berkeley. Both processors are implemented in Xilinx Vivado 2015.4 Design Suite with Virtex-7 FPGA target.

### V Conclusion

This paper details the design and implementation of a Micro controller on FPGA. The micro controller is designed to be highly flexible with customizable parameters for processor, caches and interrupts. The entire design has been tested on Virtex-7 FPGA.

### References

