Conversion of an 8-bit to a 16-bit Soft-core RISC Processor

Ahmad Jamal Salim, Sani Irwan Md Salim, Nur Raihana Samsudin, Yewguan Soo
Faculty of Electronics and Computer Engineering
Universiti Teknikal Malaysia Melaka
Melaka, Malaysia

Abstract—The demand for 8-bit processors nowadays is still going strong despite efforts by manufacturers in producing higher end microcontroller solutions to the mass market. Low-end processor offers a simple, low-cost and fast solution especially on I/O applications development in embedded system. However, due to architectural constraint, complex calculation could not be performed efficiently on 8-bit processor. This paper presents the conversion method from an 8-bit to a 16-bit Reduced Instruction Set Computer (RISC) processor in a soft-core reconfigurable platform in order to extend its capability in handling larger data sets thus enabling intensive calculations process. While the conversion expands the data bus width to 16-bit, it also maintained the simple architecture design of an 8-bit processor. The expansion also provides more room for improvement to the processor’s performance. The modified architecture is successfully simulated in CPUSim together with its new instruction set architecture (ISA). Xilinx Virtex-6 platform is utilized to execute and verified the architecture. Results show that the modified 16-bit RISC architecture only required 17% more register slice on Field Programmable Gate Array (FPGA) implementation which is a slight increase compared to the original 8-bit RISC architecture. A test program containing instruction sets that handle 16-bit data are also simulated and verified. As the 16-bit architecture is described as a soft-core, further modifications could be performed in order to customize the architecture to suit any specific applications.

Keywords—CPUSIM; FPGA; RISC processor; soft-core;

I. INTRODUCTION

Since the availability of the well-known Intel 8080 in 1972, there are numerous modern approaches that have been taken to elevate the performance, functions and features of 8-bit processors. Even until today, 8-bit processors are still going strong by being adopted as the main controller in low-end embedded system. Although the market is flooded with higher end microcontroller range such as 32-bit processors, the demands for 8-bit processors have never subsided. The driving force for the sustained interest is the lower cost, simple architecture and the suitability of the 8-bit processor to execute simple program efficiently.

Today’s 8-bit processors have incorporated advanced peripherals and features as part of its configuration such as low power consumption, sleep and wake and other on-board peripherals. However, with data limited to 8-bit wide, its operations are bound to several constraints particularly when dealing with arithmetic and logic calculation. Digital Signal Processing (DSP) calculation for example requires at least 10-bit data to ensure data reliability and computational accuracy. In this case, upgrading to a higher bit system would be the obvious choice although, the higher-end microcontroller are integrated with all sort of advanced features that the system does not necessarily required.

The term soft-core processor reflects the capability of a designer to customize the processor architecture to suit any specific application. Usually implemented in an FPGA, a soft-core processor is described in hardware description language (HDL) that enables the designer to configure the processor’s architecture by reprogramming the codes. Although there are slight performance deficit compared to the discrete processor, the ability to customize the architecture has out-weighed other undesirable factors. The designer has the direct control on the processor configuration such as memory capacity, I/O pins, instruction sets and also the data bus. Nevertheless, any modifications made to the architecture will results in a total overhaul of the processor system as all modules are tightly integrated and connected with each other.

This paper demonstrates the method in expanding the functionality of a basic 8-bit RISC processor by converting the processor’s data bus from 8-bit to 16-bit. Indirectly, the new architecture provides the capability of a 16-bit processor while maintaining the simplicity of a low-end architecture. Few other modifications are also added to the new architecture to support 16-bit operations such as multiplications. The new architecture is verified using CPUSim simulation tool [1] and also implemented on Virtex-6 FPGA

II. RESEARCH BACKGROUND

As the microprocessor world evolved to higher-end processor such as 32-bit and 64-bit architecture, the demand for 8-bit processor is still maintained albeit it existence since 30 years ago. Although some observers were contemplating the
death of 8-bit CPUs, the low-cost and easy operational features have made the processor still relevant throughout the years. Obviously the demands are focusing on low-end and non-critical applications.

Due to its simplicity and handling I/O signals, 8-bit processors are often being used as the co-processor for various control algorithm implementation. Its fast execution and stable operation enable control algorithm such as fuzzy logic [2], positioning control [3] and PID controller [4] to be implemented in embedded processors. Although the main processing part that is responsible for calculation and data manipulation are still operated by larger processors, the accompanied 8-bit processor is useful as interface to drive I/O devices to and from the main processor [5-7].

Eight-bit soft-core processors also find its place in several recent projects involving image processing [8] and multimedia applications [9]. Having a soft-core architecture have enables the processor to be expanded to accommodate more complex features especially for calculation intensive applications. In security applications, data encryption processors are employed in 8-bit processors to take advantage of the low-power configuration [10, 11].

The sector that benefits most from 8-bit processors is in teaching and learning where the 8-bit architecture is widely adopted as part of the computer architecture curriculum [12-14]. Due to its simple and basic architecture, students would be able to grasp the architecture organization before embarking on a more complex computer architecture. Further pedagogy and teaching aids based on an 8-bit architecture are developed continuously by educators to improve the learning experience in various level of education.

The use of an 8-bit processor as verification platform for new methods or algorithms implementation are usually utilized by researchers either as a stand-alone processor or integrated as part of a bigger system. The concepts of organic processors [15], non-volatile processors [16] and advanced security features [17] are all executed on top of an 8-bit architecture. The results obtained from such experiments will definitely be upgraded to higher range processors in the future.

With numerous applications and platform taken from recent reputable publications indicate that the 8-bit architecture is still relevant and contributed significantly to those systems. By modifying the architecture from 8-bit to 16-bit data bus, it will enhance the capability of the processor in handling larger data sets while at the same time maintaining the simple core architecture. The architecture’s implementation in reconfigurable platform, as soft-core processor, provides more opportunities in customizing the core to suit any specific applications.

### III. METHODOLOGY

Fundamentally, an 8-bit processor offers sufficient platform in developing low-to-medium complexity microcontroller system. Projects involving basic I/O manipulation, all-digital processing and small memory requirement would benefits from an 8-bit processing power without going through many unused features on more advanced microcontroller families.

In order to make a smooth transition from an 8-bit to a 16-bit processor, the original architecture of an 8-bit processor is referred to before modification being made. Table I shows the available specification of an 8-bit processor and the targeted 16-bit processor configuration.

#### A. Data Bus Expansion

**Abbreviations and Acronyms**

<table>
<thead>
<tr>
<th>Device Components</th>
<th>Original 8-bit RISC Processor</th>
<th>New 16-bit RISC Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data bus</td>
<td>8-bit</td>
<td>16-bit</td>
</tr>
<tr>
<td>Instruction register</td>
<td>12-bit</td>
<td>22-bit</td>
</tr>
<tr>
<td>Address register</td>
<td>5-bit addressable with 2-bit bank</td>
<td>7-bit direct addressable</td>
</tr>
<tr>
<td>Memory organization</td>
<td>Bank switching</td>
<td>Direct addressable</td>
</tr>
<tr>
<td>Status register</td>
<td>No Overflow bit</td>
<td>Overflow bit</td>
</tr>
<tr>
<td>Stack</td>
<td>3-level</td>
<td>8-level</td>
</tr>
</tbody>
</table>

Although 5-bit address point to 32 memory locations, adopting memory bank switching doubles the effective address. Extra 2-bits are added to the file register bits to form the effective address. This techniques, called memory bank switching, is efficient in expanding the memory space by using a small number of register bits but the tradeoff is that the user must initiated the specific bank in the program code to access the correct memory address. This extra code could be simplified by using program macros but nevertheless it would be cumbersome to the programmers to be consistently aware the current bank location. Therefore, for the 16-bit processor, the bank switching technique is omitted and all memory address is directly accessible from the file register bits.
D. Status Register Modification

The original 8-bit processor has a status register that consisted of carry (C), digital carry (DC), zero (Z), timeout (TO), power down (PD) and program page preselect bits. With only limited arithmetic operation such as add and subtract, calculation executed in the 8-bit processor, overflow bit is not considered to be included as part of the status register. However, as the processor expands to 16-bit data, the arithmetic operation could also include more complex calculation such as multiplication, division and multiply-accumulate. In this case, an overflow bit is introduced to provide the capability in determining the correct results especially in calculation involving signed numbers. The overflow bit is located at the 7th bit in the status register.

E. Stack Level Expansion

The new 16-bit processor also incorporated a total of 8 level stacks, an increase from 2 level stacks in the original 8-bit processor. Having more stacks could assist in more prudent program development where more active subroutines could be initiated while also providing additional local data storage and parameter passing. Subsequently, the stack pointer register is also modified to accommodate all additional stack levels.

F. Development Platforms

The modified 16-bit architecture is developed in two platforms which are CPUSim and HDL modules. CPUSim is utilized for architecture simulation and assembly test program development. Basically, CPUSim is a Java application which enables users to design simple computer CPU at the microcode level and to simulate assembly-language programs on that CPU [1]. In the architecture, all instruction sets are programmed with their own opcodes and its own operational procedures as indicated by the ISA. An assembly language test program is developed which includes the instruction sets for verification purposes. During simulation, the instruction sets execution are observed at register level on data movement from CPU registers to memory locations and vice versa. Upon successful execution, CPUSim will then provide the listing file of the test program. The listing file contains the hexadecimal representation of the program code compiled in CPUSim. The file would then be used in ROM initialization during the FPGA implementation at a later stage of this project.

For the other platform, all changes proposed are realized on hardware architecture through HDL modules. The soft-core RISC processor is divided into several modules that defined the hierarchy of the system. Most HDL modification is done at CPU modules where all registers, buses, stacks and wire connections are initiated. Instruction Decoder (IDEC) module is responsible in handling all instruction decoding and here the new instruction sets’ opcode are initialized and configured according to the new ISA. By eliminating memory bank switching procedure, the data memory access from CPU module is now more direct without the need to calculate the effective address for each command. Considerable modifications are made at ALU module whereby new operations are coded in tandem with the new instruction sets. Additional procedures are also setup to determine any overflowing results as an indicator for the new overflow bit in the status register.

The behavioral model simulation is performed on the 16-bit RISC architecture using ISim simulator. The test program (which is developed using CPUSim previously) is converted to the coefficient file (.COE) format first before it being initialized in the ROM instantiation. Behavioral model simulation includes testbenches that indicate stimulus (basically a 100MHz clock signal and reset input) for the architecture. The ISIM simulator shows the waveform of the processor execution in time sequences. The output waveform is observed for correct instructions fetch and output execution.

Later, the architecture is synthesized, placed and routed (PAR) and implemented on the FPGA board using the generated bit-stream. Utilizing ChipScope Pro features in Xilinx ISE Design Suite, all related registers are monitored instantaneously during the program execution in the FPGA board. Essentially, the ChipScope Pro Integrated Logic Analyzer (ILA) core is a customizable logic analyzer core that can be used to monitor any internal signal of the design. The ILA core includes many advanced features of modern logic analyzers, including Boolean trigger equations, trigger sequences, and storage qualification. These features will definitely improve the debugging process of the architecture design during the FPGA hardware implementation.

IV. HARDWARE SIMULATION & SYNTHESIS RESULTS

The new RISC architecture is implemented on the FPGA platform by using Xilinx Virtex-6 board. Xilinx ISE Design Suite that comprises the ISim simulator and Chip Scope Pro are utilized as the main tools in order to execute the synthesis process, place-and-route and the architecture implementation.
A test program is developed using assembly language and is assembled in the CPUSim environment. Fig. 2 shows the content of the test program and its corresponding register output during program simulation. Basically, the test program will store FF00 to memory location 10 through indirect addressing register. Then, the content of the INDF (which is the memory content of location pointed by File Select Register (FSR)) is XORed with 7FFE and the results are stored back to the location pointer by FSR. The operations involving INDF register are executed to verify the code execution when dealing with indirect addressing mode. For the next instruction, the value of INDF is added with literal value FA00. The addition’s result is checked for overflow through bit testing on status register.

It can be observed here that CPUSim also has the capability to assemble the instruction sets to its respective machine codes. Subsequently, the assembled program opcodes could be saved as listing file and it can be used to initiate the program memory module during the hardware implementation phase.

Fig. 3 shows the behavioral model simulation for the same test program adopted earlier in CPUSim. The ISIM simulator ran at normal speed according to the testbench setup and all instruction sets routine are conducted in timely manners. During the execution of the instruction sets, all results are performed properly where the instruction sets are fetched, decoded and executed as predicted. Arithmetic and logic calculations are also verified while control instructions are executed perfectly in term of overriding the program counter register to include the new program memory address. During the “addwf” instruction execution, the ALU has fetched the correct data and correct result is also produced and stored at the working register. Overflow bit is also triggered correctly, matching the CPUSim simulation result earlier.

While ISim simulation shows encouraging results, the main task for the project is to successfully implement the design in the FPGA chip. The RISC architecture is implemented in the Virtex-6 FPGA by programming the design’s bit-stream to the flash memory chip and the program is executed on power-up. The conventional way of evaluating the output results by logic analyzer is replaced by Chip Scope Pro Integrated Logic Analyzer (ILA) that ultimately speeds up the debugging process.

In ChipScope Pro, the triggered points are defined to form the waveform signals, matching the output waveform produced by ISIM simulator. Due to the large allocation of logic cells in Virtex-6, integrating the ILA core with the current 16-bit architecture is a straightforward process. The trigger sample is set to 4096 samples per triggered and the result is shown in Fig. 4. All corresponding registers have correctly record data and the overflow bit is triggered accordingly during the “addwf” instruction execution.

In term of device utilization, Table II shows the comparisons between the original 8-bit architecture and the new 16-bit architecture in term of slice registers, Look-up Table (LUT), LUT-Flip Flop pairs and the maximum operating frequency. The percentage differences are quoted to indicate changes required to accomplish the modified 16-bit RISC architecture.

With twice more of data that can be handled as compare to
its predecessor, the 16-bit architecture requires only additional 24% of LUT modules. Coupled with other features such as deeper stack level, more condition bits and added capability for instruction set optimization, the impact of an 8-bit to a 16-bit architecture conversion can be considered as minimal. There is a concern for lower maximum frequency operation which is.

V. CONCLUSION

Converting an 8-bit RISC processor to a 16-bit capability is not only about the expansion of the data bus width but also involved mostly all modules integrated in the RISC processor architecture from ALU, memory space, instruction set architecture and other internal registers. This paper has shown the potential in customization and reconfiguration of a RISC processor core that could be unlocked by having a soft-core setup for the processor’s architecture. Other than the data bus width, the RISC architecture has been modified to include few other changes that indirectly enable the processor to execute more complex operations while maintaining its efficient resources utilization. Furthermore, the modification of an instruction set’s opcode have opened up the possibility in accommodating new instruction sets and is a precursor to the concept of application-specific instruction set processor (ASIP).

REFERENCES