International Journal of Research in Engineering, Science and Management
Volume-2, Issue-5, May-2019
www.ijresm.com | ISSN (Online): 2581-5792

RFU based CPLD Compiler Approach

S. K. Viswanath[1], M. Dhanaraj[2], C. Y. Santhosh[3], Sini Anna Alex[4]
[1,2,3]*Student, Dept. of Computer Science and Engg., M. S. Ramaiah Inst. of Tech., Bangalore, India*
[4]*Assistant Professor, Dept. of Computer Science and Engg., M. S. Ramaiah Inst. of Tech., Bangalore, India*

***Abstract*: We propose a chain compilation in which the compiler is not limited to a fixed pre-defined instruction set here the application can generate application specific custom instructions and synthesize them in FPL. We compare the FPGA and CPLD respectively and the CPLD core that implements the RFU based on the Philips XPLA2 architecture. Why XPLA2 instead of FPGA. Also here we are proposing about 40% more speeds for the encryption algorithms using the RFU-extended CPU can achieve. When compared to the standard CPLD CPU core alone.**

***Keywords*: FPGA, CPLD, RFU, FPL.**

1. Introduction

Microprocessors have a fixed limited instruction set on to which application program must be mapped by the compiler. instructions are hard wired and executed as ALU's function units. Performance could be considerably improved the compiler could define custom instructions specific for the application that needs to be run. in order to allow this microprocessor should feature reconfigurable functional unit able to implement and execute the different custom instruction at compile time.as a role RFU are implemented in FPL (Field programmable logic). RFU opens new degree of freedom, since it can define custom instructions on per application at hand.

The potential of the possibility developed a compiler driven approach for integrating RFU based custom instructions. The approach targets embedded systems. compiler time analysis aims at encoding multiple custom instructions in a RFU configurations. the main objective is to reduce the FPL overhead reconfigurations without the complex run time partial reconfigurations schemes as used in previous work which is too costly in embedded systems. We are utilizing CPLD architecture as our RFU which makes way for numerous advantages with respect to FPGA.

2. Related works and applications

A. FPGA working

Its type of device that is used in electronic device. FPGA is semiconductor device which contain programmable logic blocks and interconnection circuits. In this algorithm hardware accelerators, alternatively speed-up the performance and lower the power consumption. The main aim of this is conversion of C code to the static dataflow machine designed using FPGA. The result of this is an newer alternative of the development of hardware accelerators, with good performance and lower power consumption. The dataflow based accelerators are a solution for the increasing the computational power. The FPGA are very highly customizable, and this provides the perfect hardware development for different needs, beyond and also have a very low power consumption the development of FPGA is also responsible for the resumption of research in dataflow architecture.

Working:

- Data flow computation.
- Dataflow graph generation.

In the data flow computation, the node is acts as an element and the arc is a connection between two elements. A data and control bus are used to perform the communicational operations and performing a traditional operator like add, sub, mul, div etc.

In the data graph generation, the software is usually made for using a specific compiler and ANSI C. In this the intermediate representation is generated. This is based on three address codes(3AC). after the IR generation, a dataflow graph is generated for this and each operator output is computed and verified. Dataflow application on each run the compiler must insert data in a FIFO structure.

B. Application

- FPGAs have been accepted quickly over the past few years
- There are large number of application like random logic SPLDs device controllers and filtering
- They provide solution for video surveillance, motor control, machine vision.
- They are used in custom computers
- It provides a combination of highly parallel low-cost computation.

3. The architecture

Here we develop a method and RISC micro architecture for improving the processes performance in embedded systems applications by the custom RFU based instructions avoiding the disadvantages mentioned in section [2].

1. we reduce the overhead caused by the latency reconfigurations of FPL resources, here we still are able to avoid the run time partial reconfigurations which hence in increased hardware cost and complexity.
2. All the hardware synthesis is hidden from the

programmer by the means of a smart compiler.

Our proposed theory consists of encoding multiple custom instructions in a single RFU thus decreasing number of times RFU must be reconfigured. RFU instruction set is defined as a register-register operation with one specific opcode which is of six bits. below we illustrate standard RISC register-register encoding format.

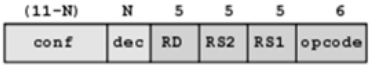

Fig. 1. Encoding format of RFU instructions

Our architecture feature one extra functional unit in parallel with ALU. The CONF signals input reconfigurations control unit, which is attached to CPLD core which controls the loading of different configuration codes this is what makes our method different, moreover it is the DEC signals that directly go to the CPLD core and define which of the custom multiple instruction which are encoded in the given configurations is to be executed. The CPLD configuration itself features on instructions decoder capable of interpreting the DEC signals and the pipeline register is added to buffer the CONF and DEC in between pipeline stages.

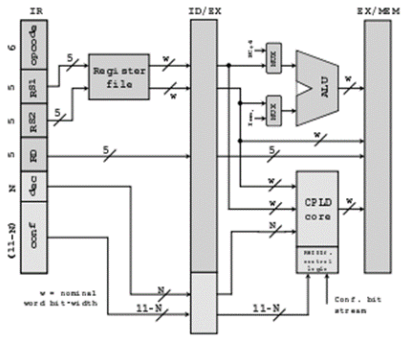

Fig. 2. Proposed architecture

Fig. 2, shows the possible block representation of our architecture, multiplexers driving pipeline registers controlled by opcode bits are omitted. Only the relevant signals for register-register operation are shown, the RFU instruction never incorporates the immediate value or addresses but its operands must be derived after the bypassing objects as mentioned in fig. 2.

4. RFU Based CPLD

To define the embodiment for our CPLD we will consider N=4 from fig. 1 and fig. 2 and w=32 from fig. 2 which is 32-bit architecture fig 3 shows the CPLD core it is the stripped down version commercial SRAM based XPLA2 device. To logic blocks run the 36 input signals each from LZIA (low zero power interconnect array) which is cross points switch with very low power consumption. The inputs for LZIA come from to 32 input pins taking the signals DEC. Each logic block as 16 output pins which total 32 output bits of the result operand the internal block diagram of logic is shown in fig. 4.

The addition of the PAL and PLA arrays in logic block makes way for building complex logic functions with single pass through array each of the output pin as four dedicated product terms from the Pal array which is connected to it so the total of 36 PT's can be used to run a single result bit. PLA array provides extra needed PT's without taking or stealing terms from the neighboring pins. If PT's were to be taken from the neighboring pins this pins cannot be used.

The PAL/PLA combination makes way for PLA PT sharing among different macro cells. This increases the density effectiveness of the device and allows for larger and more complex functions implementations. Additional features of this CPLD architecture is its predictable timing model the delay depends only on whether or not PT's from PLA are used

- In case only Pal terms are used the delay associated TPD_PAL includes a pass through LZIA and PAL.
- Second term from PLA are used the delay associated called TPD_PAL as a pass through the LZIA and PAL.

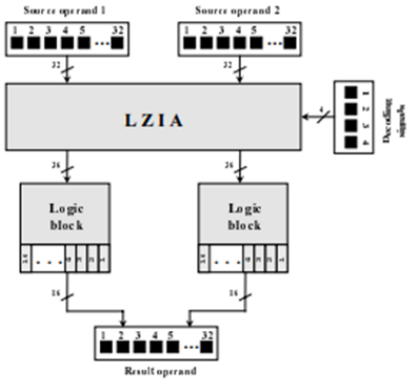

Fig. 3. CPLD core

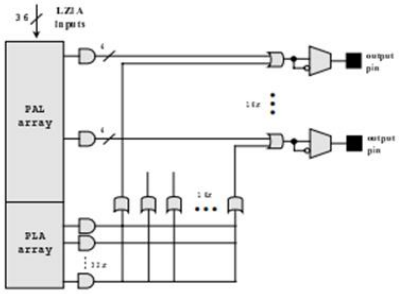

Fig. 4. Logic block

We can now see the advantage of using regular CPLD structure in RFU however complex its function needs to be provided it fits in the resources the timing is limited to TPD_PLA, while using more PT's from the PLA the complexity of the design increases without effecting the timing this is important when taking in to account that an arbitrary number of custom instructions must be encoded in to a single configuration and RFU should execute within the stage of RISC

pipeline this type of regularity in timing model cannot be achieved through regular FPGA architecture. We estimate RFU will take 4mm^2 of silicon in C075 this compare to Philips PR3930MIPS processor in the same 0.35micro-meters processes RFU is very meager investment in silicon which is targeted more cost effective for certain embedded applications larger than caches or higher clock frequencies.

5. Conclusion

We have presented a new compilation an architecture approach for RFU extended reconfigurable RISC cup's this method targets embedded systems and also aims at being more cost efficient in certain application then increased cache size or clock speed. Multiple instructions automatically generated by the application code of a smart compiler this could be encoded in a single RFU configuration the instruction decoding part was performed at a later pipelining stage, during execution in RFU by this method we achieved the objective of reducing the reconfiguration overhead of RFU based CPU's, it has allowed for optimized usage of programmable logic recourses. This complex and costly schemes are hard to justify in the applied embedded domains where simplicity and reliability

This approach helps in efficient implementation of bit level manipulation which are common in encryption algorithm by achieving speeds more than 40 % compare to normal RISC without RFU. This opens a new degree of freedom for encryption algorithm writers. They need not restrict themselves to blocks of bit manipulation but now freely use bit log computations.

References

[1] W. C. F. Cameron and Kirk, "The Green500 List: Encouraging Sustainable Supercomputing," Computer, vol. 40, no. 12, pp. 50–55, 2007.

[2] J. Teifel and R. Manohar, "An asynchronous dataflow FPGA architecture," in *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1376-1392, Nov. 2004.

[3] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. Stat. Mech. Theory Exp., vol. 2008, no. 10, P. 10008, Oct. 2008.

[4] A. Duran, J. Corbalan, and E. Ayguad ´ E, "Evaluation of OpenMP ´task scheduling strategies," in International Workshop on OpenMP. Springer, 2008, pp. 100–110.

[5] J. Ahn and K. Choi, "Isomorphism-Aware Identification of Custom Instructions with I/O Serialization," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 34-46, Jan. 2013.