Reconfigurable Computing: Its Concept and a Practical Embodiment using Newly Developed Dynamically Reconfigurable Logic (DRL) LSI

Masakazu Yamashina and Masato Motomura

Silicon Systems Research Laboratories
NEC Corporation
Kanagawa, Japan
Tel: +81-42-771-0746
Fax: +81-42-771-0881
E-mail: {yamasina, motomura}@mel.cl.nec.co.jp

Abstract - This paper first outlines a broad range of reconfigurable computing research activities from a perspective of system LSI designs. Then, the paper focuses onto dynamically reconfigurable logic (DRL) LSI, a prototype chip that we developed to evaluate the reconfigurable computing concept. Through its ability to exchange hardware contexts quickly, this chip can accelerate media/communication applications with customized hardware configurations, yet maintaining scalability towards varying application sizes.

I. Introduction

Reconfigurable computing is an approach to solve a given problem by using a hardware whose structure is appropriately customized to the application. There is a fairly long history of research activities in this broad category, that are called under different names, such as configurable computing [1][2], reprogrammable computing [3], custom computing [4], or virtual hardware [5]. Majorities of those past works have concerned with establishing a general-purpose computing framework that subsumes a conventional computing approach in some computation or data intensive application domains [4].

Recently, reconfigurable computing is gaining renewed interests from a system LSI design perspective [6]. This is because a *reconfigurable computing IP core,* that changes its configuration to suit the application, can fill a wide gap between programmable and hardwired IP cores. Here, the wide gap exists both in performance and programmability: a hardwired IP core can achieve orders of magnitude better performance than a programmable IP core, while the latter can perform virtually any function when the former handles only the task it is designed for. Reconfigurable computing IP core may become a key component for software-hardware co-design of a system LSI, through its ability to achieve both hardware performance and software programmability.

In this paper, we first overview concept and potential benefits of reconfigurable computing. We then introduce newly developed dynamically reconfigurable logic (DRL) LSI, which is a proof-of-concept prototype for a reconfigurable computing IP core in the near future.

II. Reconfigurable Computing

A. Concept

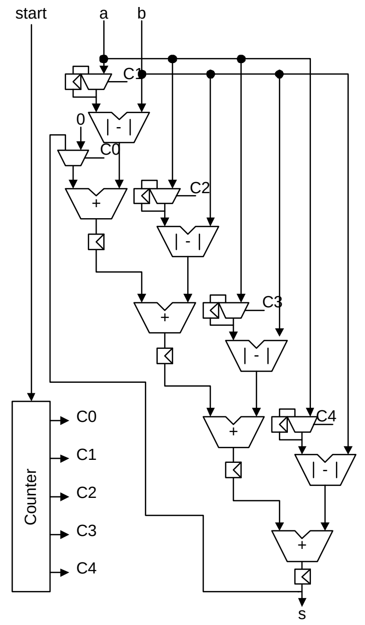
Figure 1 explains the difference between conventional and reconfigurable computing with using a simplified motion vector detection routine, often used in video compression algorithms, as an example. In a conventional computing system, an original source program [Figure 1(a)] is compiled into a machine code [Figure 1(b)]. This machine code consists of a stream of instructions that are pre-defined as a target of compilation. In case of reconfigurable computing, the same program is compiled into a hardware configuration [Figure 1(c)], to which data and control flows of original source program is directly mapped. This hardware configuration is *instantiated* when the routine is to be executed. Important observations obtained from this simple example are the following:

```
s=0;
for(j=0; j<256; j+=N) {
  for(i=0; i<16; i++) {
    s=s+abs(a[j+i]
         -b[base+j+i]);
  }
}
```

(a) Source program

```
j <- 0              :L0
  s <- 0
   i <- 0           :L1
    h <- i+j        :L2
    a <- A[h]
    b <- B[base+h]
    i <- i+1
    d <- |a-b|
    s <- s+d
    branch L2 if(i<16)
   j <- j+N
   branch L1 if (j<256)
 .
 .
```

(b) Machine code



(c) Hardware configuration

Figure 1. Comparison between conventional and reconfigurable computing approaches.

- The conventional approach solves the problem in a *temporal* manner with using sequence of instructions, while the reconfigurable approach does this in a *spatial* manner with using mutually interconnected operation units [7][8].
- In the conventional approach, register and instruction sets are the two important key components that define the interface between a compiler and hardware.
- In the reconfigurable approach, programmable logic blocks and programmable interconnections, which are provided in a reconfigurable computing fabric (such as an FPGA), define the interface between a *compiler* and a hardware.

It is important to note that, despite those differences, both the conventional and reconfigurable computing approaches are programmable: the compilers generate the bits that specify how a given problem is to be solved [9]. Principal distinction is that the "bits" in reconfigurable computing case controls detailed hardware configuration that has not been exposed to compilers conventionally [10]. Because of this, a compiler for reconfigurable computing (RC compiler) carries characteristics of both conventional compiler and conventional LSI design tools [7].

B. Impact on System LSI Design

There is a growing concern in the industry that we hit more difficulties as the technology allows us to integrate more transistors on a chip. The difficulties lie in various aspects, such as 1) how to utilize huge number of transistors, 2) how to design them, 3) how to achieve good yields, 4) how a chip is tested, and so on. We have already argued that reconfigurable computing IP core is attractive from a point of view of software-hardware co-design. It also has many other attributes that are potentially important for solving the problems mentioned above.

1) *Transistor utilization:* A reconfigurable computing IP core can turn increased number of transistors directly into improved performance, by enlarging the array size of its fabric. In conventional processors, most of the transistors are devoted to cache memories (and to complex instruction issue logic), which contribute to the performance only in a saturated manner.
2) *Design:* It is easier to design than hardwired IP cores because of its regular array structure. Also, since it is used repeatedly in various system LSIs, costs for an optimum design can be compensated more easily.
3) *Yields:* The regular array structure makes it easier to tune fabrication process to fit with its geometrical characteristics. It is also possible to include redundant cells to improve the yields by using after-fabrication replacement. These techniques are analogous to the techniques used to improve the yields of memory LSIs. Moreover, it may become possible to devise self-repairing mechanism by taking the advantage of redundant array structures [11].
4) *Testing:* It even becomes possible to include built-in test mechanism within the regular array structure in order to enhance the testability. A similar technique is already employed in module based arrays [12].

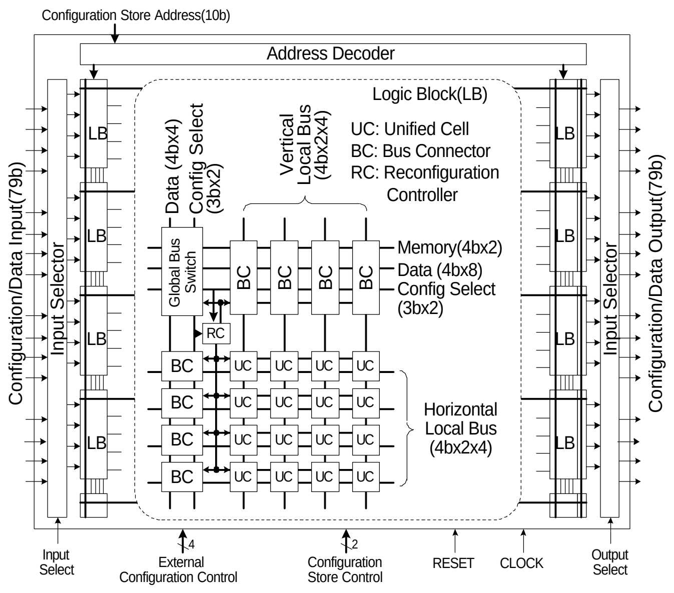
In essence, since a reconfigurable computing IP core is of general purpose, the problems regarding "how it is designed and manufactured" and the problems regarding "how it is used" can be decoupled cleanly. (These problems are inter-mixed in case of an application-specific LSI design, which complicates the design process.) The LSI design tools now deals with the former, while the RC compiler takes the responsibility of the latter.

C. Applications

Target application areas of a reconfigurable computing IP core is media-centric and network-centric processing, to which conventional programmable IP cores, such as a μP or a DSP, often fail to deliver sufficient performance. This is because their characteristics, *e.g.,* abundant yet irregular parallelism, narrow-bit-width and/or variable-length data elements, stream-based processing, do not fit well with a register/instruction-set based, fixed word-length architecture.

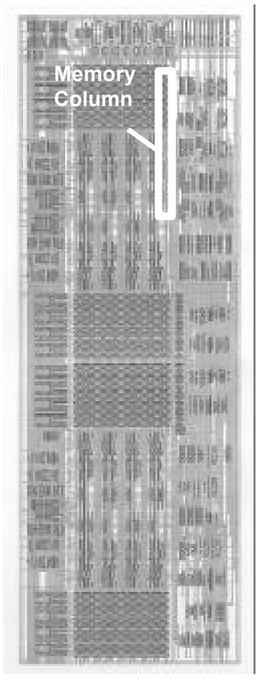
D. Required Architecture Improvement

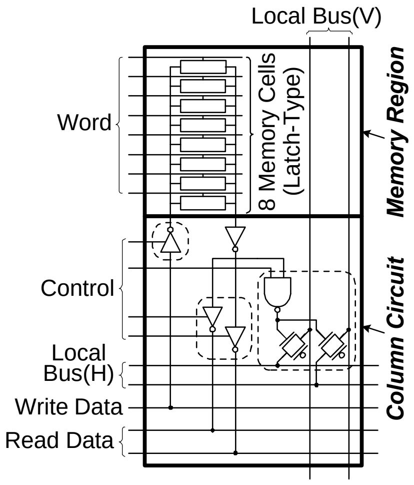
The performance improvements of reconfigurable computing come from the fact that it solves an application spatially with a customized hardware. Its programmability, on the other hand, is determined by how reconfiguration is conducted quickly and flexibly. Conventional FPGAs have not focused on this "reconfigurability" issue much, because they have been used mainly for emulation and prototyping purposes. In order to make a reconfigurable computing approach practical, the "reconfigurability" of the conventional FPGA architectures should be improved significantly [13][14].



Configuration Store Address(10b)
Address Decoder
Logic Block(LB)
UC: Unified Cell
BC: Bus Connector
RC: Reconfiguration Controller
Data (4bx4)
Config Select (3bx2)
Vertical Local Bus (4bx2x4)
Memory(4bx2)
Data (4bx8)
Config Select (3bx2)
Global Bus Switch
RC
BC
UC
LB
Horizontal Local Bus (4bx2x4)
Configuration/Data Input(79b)
Input Selector
Configuration/Data Output(79b)
Input Selector
Input Select
4
External Configuration Control
2
Configuration Store Control
RESET
CLOCK
Output Select

Figure 2. DRL prototype LSI.





(a) Unified cell (b) Memory column circuitry

Figure 3. An unified cell and a memory column circuitry.

III. Dynamically Reconfigurable Logic LSI

As a vehicle for proving the reconfigurable computing concept and evaluating application performance, we have designed and fabricated a dynamically reconfigurable logic (DRL) prototype LSI [15][16]. This chip integrates 5.1M transistors on 10.1mm square die with using 0.25μm CMOS technology. It has eight on-chip configuration contexts, and can switch from a context to the others within a single operation cycle.

A. Architecture Overview

Figure 2 shows an overall block diagram of the DRL prototype. It contains a 4 x 12 array of logic blocks (LBs), an address decoder, and input/output selectors. An LB comprises a 4 x 4 array of unified cells (UCs), a global bus switch, bus connectors, and a reconfiguration controller. Horizontal and vertical global buses provide chip-level interconnections among LBs within a row and a column. Each UC is directly connected to its four nearest neighbor UCs through two sets of horizontal and vertical local buses, without insertion of a switch matrix. 79b input/output ports for configuration contexts and operands or results are located on the left/right side of the chip, respectively.

B. Unified Cell

A plot of a unified cell (UC) is shown in Figure 3(a). The UC is 114-μm by 337-μm large, and it contains 4,552 transistors. It mainly consists of 4 x 4 matrix of memory columns (MCs), which are layouted in a pitch-matched

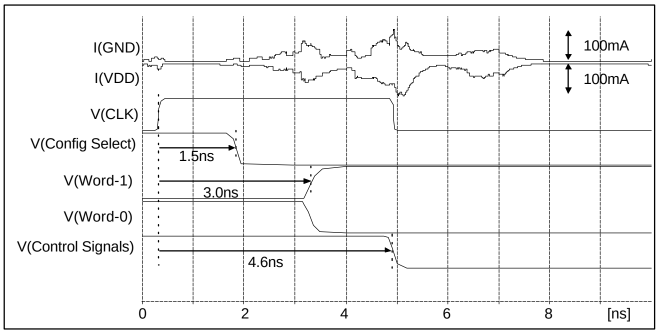


Figure 4. Simulated waveforms of dynamic reconfiguration.

manner. An UC can be configured into 4-input/1-output arbitrary logic circuits, 4 x 4 interconnection matrix switches, etc., with using a conventional look-up-table scheme.

An MC circuitry is depicted in Figure 3(b). It consists of a memory region and a column circuit. The eight memory cells in the memory region stores eight configuration contexts, from which *Config-Select* (see Figure 2) activates a current context. Here, a single configuration context contains all the necessary information to configure an UC. Thus switching between those configuration contexts results in ultra-fast reconfiguration.

C. Dynamic Reconfiguration

Figure 4 shows simulated waveforms of dynamic reconfiguration. The simulation is conducted by running a transistor-level power simulation tool on a back-annotated netlist after the layout design is completed. Here, we simulate dynamic reconfiguration of a single LB from a sample context 0 to a sample context 1 in a 10-ns cycle time. In Figure 4, the transition of *Config-Select* at an UC input takes 1.5 ns from the clock edge. It takes another 1.5 ns to switch an active word line in an MC from Word-0 to Word-1. The new context is read out through the column circuits in the MC circuitry, which generate various control signals. The critical path delay of dynamic reconfiguration is 4.6 ns. It is more than five orders of magnitude faster than reconfiguration latency of conventional FPGAs

Figure 4 also shows current dissipation during the dynamic reconfiguration. Average current is 16.5 mA, which results in 2.0-W power dissipation for whole-chip reconfiguration at 100-MHz clock frequency. Since this calculation assumes whole-chip reconfiguration to take place in every clock cycle, this power dissipation number is unrealistic. Suppose an application conducts reconfiguration once in every hundred cycles, the power dissipation due to dynamic reconfiguration is only 20-mW.

D. Application Examples

We are examining various communication-processing applications on the developed DRL prototype chip. The evaluation results on data encryption standard (DES) algorithm has been already reported [16]. In comparison to a conventional high-speed microprocessor, the DRL has achieved an order of magnitude better performance with two orders of magnitude better energy utilization. Other evaluation results will be reported elsewhere.

In order to show how dynamic reconfiguration is effectively utilized, Figure 5 illustrates a packet-processing example. As is well known, a network packet has multiple headers in front of its payload, each of which corresponds to each layer in a protocol stack. Not only its bit-length and field organization but also how it should be analyzed are different from a header to header. A DRL can process those headers with customized hardware configurations. Since how to handle a header in one layer depends on the results of previous header analysis, quick adaptation of hardware by using DRL's rapid reconfiguration mechanism is indispensable.

IV. Conclusion

We have examined the feasibility of reconfigurable computing through the design and evaluation of a DRL prototype chip. Our preliminary evaluation results are quite positive: a DRL can successfully achieve both hardware performance and software programmability.

Major obstacle for making reconfigurable computing truly practical is the lack of efficient RC compiler. As is already discussed, the RC compiler is something in between conventional compilers and conventional LSI design tools. As such, it will become an important and challenging research target for researchers in the LSI CAD field, including high-level synthesis [17], logic synthesis, floor-planning, and place and route tools.

Acknowledgements

We appreciate Mr. Koichiro Furuta and Mr. Taro Fujii for their indispensable efforts they have devoted in the DRL prototype LSI development. We would like to appreciate Mr. K. Wakabayashi for the stimulating discussions on RC compilers. We also would like to thank Dr. Masao Fukuma and Dr. Takao Nishitani for their generous understanding and continuous encouragement towards our work.

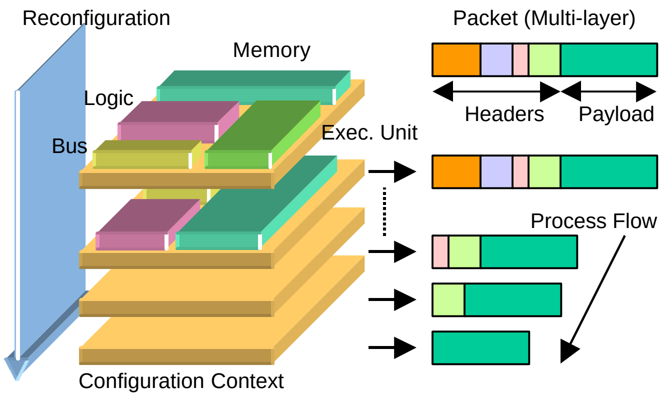


Figure 5. Application example: packet processing.

References

[1] J. Villasenor and W. H. Mangione-Smith, "Configurable Computing," In Scientific American, Vol. 276, No. 6, pp. 54--59, June 1997.

[2] W. H. Mangione-Smith, et al., "Seeking Solutions in Configurable Computing," In IEEE Computer, Vol. 30, No. 12, pp. 38--43, December 1997.

[3] S. Hauck, "The Roles of FPGAs in Reprogrammable Systems", In Proceedings of the IEEE, Vol. 86, No. 4, pp. 615--638, April 1998.

[4] D. Buell, J. Arnold, and W. Kleinfelder, "Splash2: FPGAs in a Custom Computing Machine", IEEE Computer Society Press, 1996.

[5] X. Ling and H. Amano, "WASMII: An MPLD with Data-Driven Control on a Virtual Hardware, "In Journal of Supercomputing, Vol. 9, No. 3, pp. 253--276, 1995.

[6] W. B. Andrew, et al., "A Field Programmable System Chip which Combines FPGA and ASIC Circuitry," In Proceedings of the IEEE 99 CICC, pp. 183-186, 1999.

[7] A. DeHon, J. Wawrzynek. "Reconfigurable Computing: What, Why, and Design Automation Requirements?" In Proceedings of the 1999 Design Automation Conference, pages 610--615, June 1999.

[8] A. DeHon, "Trends Toward Spatial Computing Architectures," In ISSCC Digest of Technical Papers, 21.2, Feb. 1999.

[9] M. J. Alexander and M O'toole, "Implications of Reconfigurable VLSI in a Globally Networked World: Delivering Hardware Over the Net," Washington State Univ. Technical Report, EECS-97-003, Aug. 1997.

[10] W. Lee, et al., "Space-Time Scheduling of Instruction-Level Parallelism on a Raw Machine, " In Proceedings of the ASPLOS-8, October 1998.

[11] A. Shibayama et al., "An Autonomous Reconfigurable Cell Array for Fault-Tolerant LSIs," In ISSCC Digest of Technical Papers, 14.4, Feb. 1997.

[12] http://www.lightspeed.com/

[13] M. Motomura, et al., "An Embedded DRAM-FPGA Chip with Instantaneous Logic Reconfiguration," Proceedings of Symposium on VLSI Circuits, pp. 55-56, Jul. 1997.

[14] S. Trimberger, et al., "A Time-Multiplexed FPGA," In Proceedings of Symposium on FCCM, pp. 22-28, Apr. 1997.

[15] T. Fujii, et al., "A Dynamically Reconfigurable Logic Engine with a Multi-Context/Multi-Mode Unified-Cell Architecture," In ISSCC Digest of Technical Papers, 21.3, Feb. 1999.

[16] T. Fujii, et al., "A 0.25-μm CMOS, 5.1-M-Transistor, Dynamically Reconfigurable Logic Engine (DRLE) LSI," In Proceedings of Cool Chips II, pp. 51-63, Apr.1999.

[17] K.Wakabayashi, "Cyber: High Level Synthesis System from Software into ASIC," in High Level VLSI Synthesis, edited by R. Camposano and W. Wolf, Kluwer Academic Publisher, pp. 127-151, 1991.