Customizable Domain-Specific Computing (June 2015)

Kun-Hua Huang, Department of Electronics Engineering, National Chiao Tung University No. 1001, University Road, Hsinchu 300, Taiwan.

Abstract - The computing technology entered the era of parallelization, with tens to hundreds of computing cores integrated in a single processor, and hundreds to thousands of computing servers in a warehouse-scale data center. To make the computing faster and more efficiently. Customizable domain-specific computing is a new method to design the computing system. The domain-specific customization integrate reconfigurable hardware on to the chip. The majority of reconfigurable hardware accelerate the specific computing algorithm. In this paper, we show how the reconfigurable hardware work. Next, we analysis the customizable domain-specific computing system. We also propose a way to significantly reduce the chip area for the customizable domain-specific computing system.

Keywords: Customizable system, domain-specific customization, CPUKH, architecture design, FPGA,

I. INTRODUCTION

IN the recent decades, computer technology has grown exponentially. To speed up the computing ability, the integrated circuit manufacturing plays an important role to accelerate the operating frequency [1]. With the limit of the integrated circuit manufacture technology such as power and frequency [2], the computing technology entered the area of parallelization, with tens to hundreds of computing cores integrated in a single processor, and hundreds to thousands of computing servers in warehouse-scale data center [3].

The warehouse-scale data center have highly computing ability in parallel and general purpose computing [4]. However, there are some weakness such as performance, power, heat dissipation, space and cost [3]. The computing efficiency of warehouse-scale data center is able to improve. There is an opportunity on the domain-specific customization [5]. The majority of domain-specific customization design the hardware that fit to the domain-specific algorithm, make the algorithm computing parallel and more efficiently [6][7]. The customizable computing means the computing component is designed specific for special computing. Thus, in domain-specific computing domain, there is a method called reconfigurable hardware [8][9]. As the name of the reconfigurable hardware, it means the hardware can be configure or be programmable. The reconfigurable hardware is useful in develop the digital circuit design and also useful in digital system design. As the advantage of the reconfigurable hardware, the customizable domain-specific computing grows up nowadays. It make lower cost, quickly and flexibly to implement the dedicated circuit for specific computing [10][11].

In reconfigurable hardware domain, there are many technology to implement the hardware. Such as Programmable Logic Device (PLD) [12], the PLD is an electronic component used to build reconfigurable digital circuit. PLD has an undefined function at the time of manufacture. Another reconfigurable hardware technology called Programmable Logic Array (PLA) [13]. PLA has a programmable AND gate array, which links to a programmable OR get array, which can them be conditionally complemented to produce and output. There is also a reconfigurable hardware technology called Programmable Array Logic (PAL) [14]. The PAL have arrays of transistor cells arranged in a "fixed-OR, programmable-AND" plane used to implement "sum-of-products" binary logic equation for each of the outputs in terms of the inputs and either synchronous or asynchronous feedback from the outputs. The other reconfigurable hardware technology called Generic Array Logic (GAL) [15], GAL has the same logical properties as the PAL but can be erased and reprogrammed. Another reconfigurable hardware technology called Complex Programmable Logic Device (CPLD) [16][17]. As the name of CPLD, the complex PLDs contain the equivalent of several PALs linked by programmable interconnections, all in one integrated circuit. In recent decade, one of the reconfigurable hardware technology called Field-programmable Gate Array (FPGA) [18]. FPGA has plays an important role in the reconfigurable hardware technology [19]. The difference between FPGAs and CPLDs is that FPGAs are internally based on Look-up tables (LUTs) whereas CPLDs form the logic functions with sea-of-gates [16][17].

Using FPGA as reconfigurable hardware for customizable domain-specific computing let the design more flexible and more efficiently [20]. Therefore, in this paper, we will discuss the FPGA technology and then discuss how the domain-specific customization working. Also, we will discuss how to decrease the area and make the customizable design more efficiently.

Kun-Hua Huang is with Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan.(e-mail: kunhuahuang@hotmail.com).

II. PROGRAMMABLE TECHNOLOGY

For domain-specific customization purpose, Field programmable Gate Array play an important role on customizable computing technology [21]. In the digital circuit implementation media, As the Figure (1) FPGAs consist of programmable logic, memory and multiplier blocks, surrounded by a programmable routing fabric that allows block to be programmability interconnected [22]. In this architecture, the FPGA can simulate most of the digital circuit device. Generally, the FPGA helps designer to evaluate and testing the digital circuit design and also testing the digital system design. Also, the already programmable FPGA can be sells as products. Some developer quickly develop the system and sell out without manufacture the circuit into integrated manufacturing. With fewer production, the FPGA get the benefit of low cost.

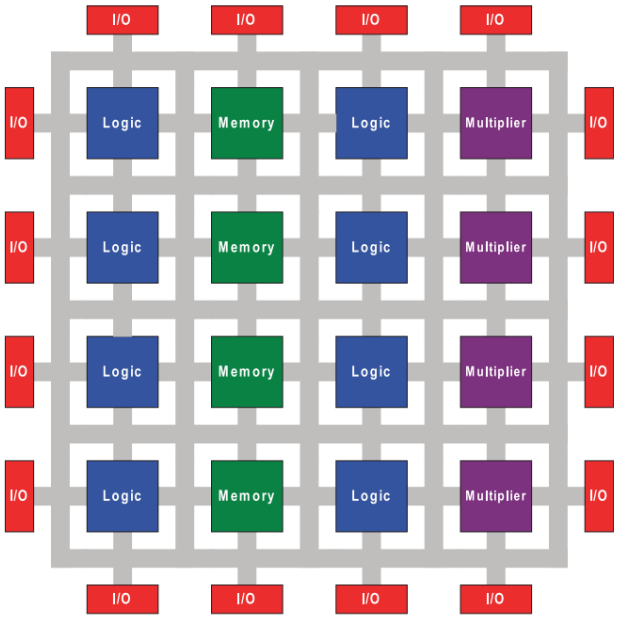

Figure (1)

A. *Introduce of Programmable Technology*

There are multiple programmable technology for FPGA. However, the SRAM programmable technology has become the primary way to control the programmable switches. The lookup-table (LUT) consists two cells: SRAM cells and mux cells. As the Figure (2), the SRAM cells are used to store the data in the lookup-table (LUTs) to implement the logic function [23][24]. And the mux cells is to select the stored data from the SRAM cells. In this LUT technology, the logic can be programmable again and again. And the cell use of standard CMOS process technology. This means the LUT implementation grow up with the Digital integrate circuit technology.

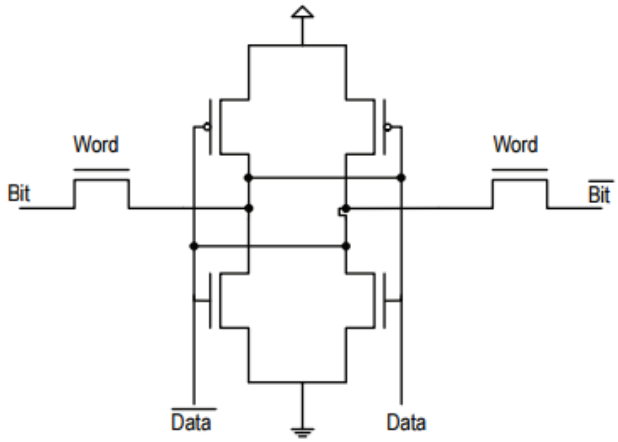

Figure (2)

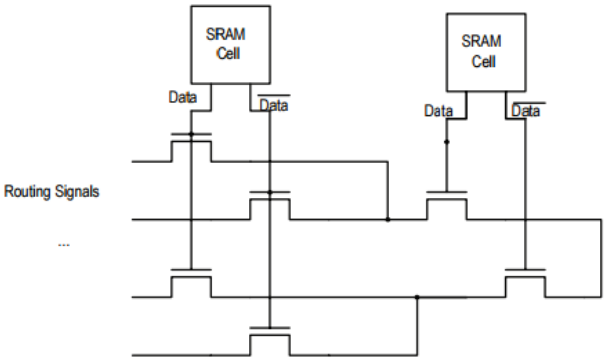

Figure (3)

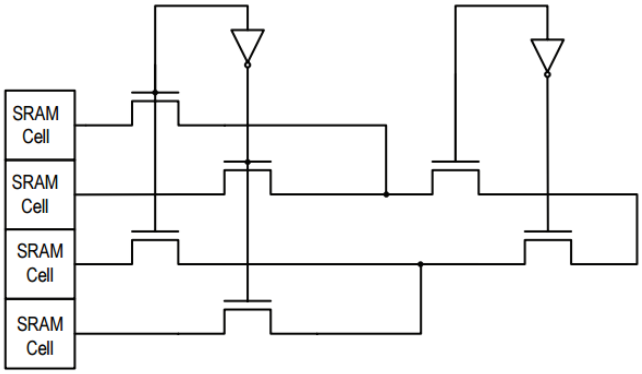

Figure (4)

B. *Logic Block Architecture*

The logic gate can be transform into truth table [25]. As using truth table method, the table stored into the SRAM cell. Then, as Figure (3) the MUX cell select the SRAM and get the value that match to the table. The two cell combine together as Figure (4).

As the figure, the basis logic block is an SRAM functioning as a lookup-table (LUT). The truth table can be stored in 2^k x 1 SRAM for a K-input logic function.

For instance [22], a logic function f - a + bc, can be made as Figure (5). The logic implementation using a three-input LUT, then the SRAM should be made as figure (6).

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Figure (5)

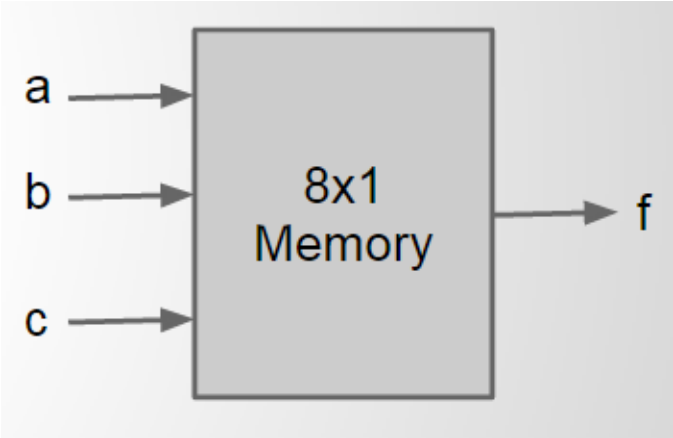

Figure (6)

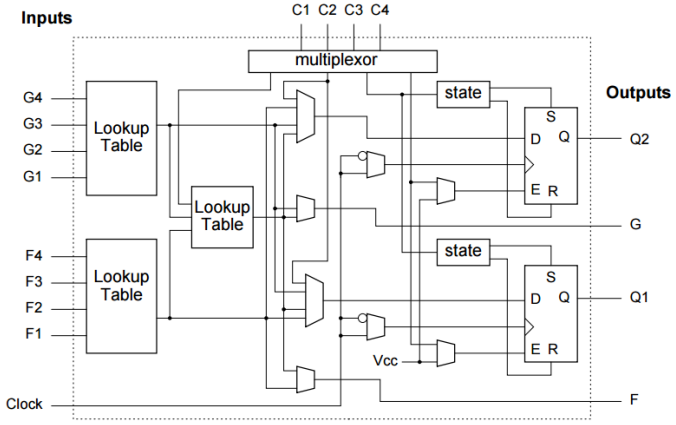

Figure (7)

C. *Configure Logic Block*

In the Xilinx 4000 series configurable logic block (CLB) [26]. As the figure (7), it contains two four-input LUT's feeding into one three-input LUT. This architecture is a trade-off for the design such as delay time and area. In this architecture, the delay time decrease but the three-input LUT maybe unused [27]. Because of these connections are significantly faster than any programmable interconnections since no programmable switches are used in series.

D. *Sequential Logic*

In digital circuit, the sequential logic is used to construct finite state machines, a basic building block in all digital circuitry.

To do the Sequential Logic can implement by CLB as Figure (7) [27][28]. There are two D Flip-Flop cell in the CLB. To use the D Flip-Flop, the wiring control by multiplexer. The wiring can be programmability connected D Flip-Flop of the two four-input LUTs, three-input LUT or from the outside interconnects. In this structure, the delay time can significant decrease. Because of less wiring which the logic data can be quickly pass the LUT to Flip-Flop without pass through the long wiring. The longer the wiring, the resistance and capacitance become stronger. The delay time will exponentially grows.

E. *Routing Architecture*

The routing architecture is important for connection between each block. In Xilinx 4000 series routing architecture, there are two layer routing interconnection [29][30][31].

The first layer is called Single-Length Lines. The majority of the first layer is used to make connection in local area. As the figure, Each X indicates a routing switch. The Switch Matrix consists of six routing switches, each wiring segment can connect thought routing switches to three other wiring segments.

The second layer is called Double-Length Lines. This layer is similar to the Single-Length Lines that the signal passes through less series of resistance in travelling the same distance.

These two layer design make the wiring more efficiently. And solve the unexpectedly long distance wiring.

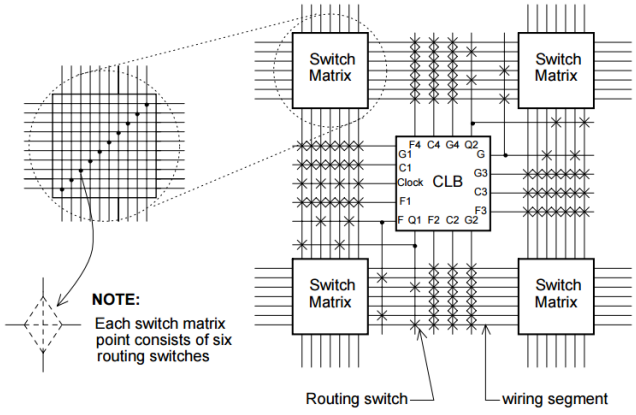

Figure (8)

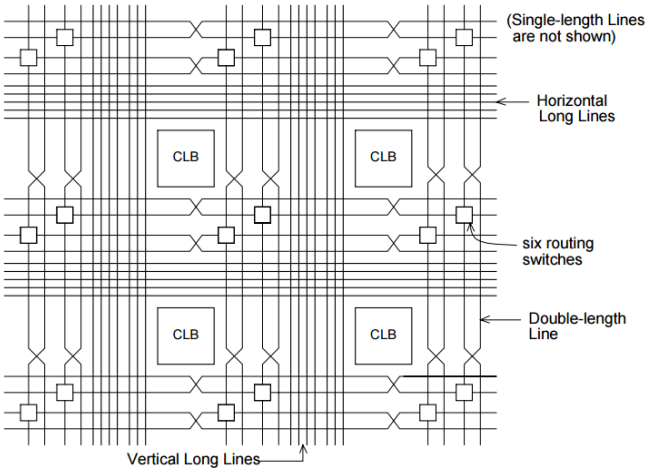

Figure (9)

F. Summary of FPGA

In SRAM-based FPGA, the CLB consists LUTs and D Flip-Flops that can be support combination logic and sequential logic computing. The routing line can support signal passing to the other block.

The reconfigurable hardware such as FPGA can simulate most of the digital circuit and digital system. In this hardware architecture, the circuit can be programmable.

In this reason, the customizable domain-specific computing can be built on FPGA environment. This way, the design can be more flexible and programmable that we can customize the specific computing algorithm on the FPGA.

III. Analysis of Customizable Domain Specific Computing

In FPGAs digital domain-specific processing applications such as Image, Video, audio and speech processing, error control… etc. These applications can be accelerate on FPGA. In "Customizable domain-specific computing" paper shows the data that FPGA provided an 11 x speedup on Biharmonic registration algorithm. On 3D median denoising filter algorithm can provided a 1000 x speedup [3].

FPGA have strength on computationally intensive and parallel computing. Unfortunately, FPGAs not suitable for irregular operation like data-dependent open loops or data transfer applications [32]. These kinds of irregular operation part needs a large chip area on FPGA while they can be easily done by microprocessors.

A complete customizable domain-specific system must consists Irregular operations and complicated computation [33]. As previous talk, the Irregular operations is best match to microprocessors architecture [32]. The complicated computation can be implemented more efficiently by FPGA [32].

In this way, using a reconfigurable hardware together with a microprocessor, both of the irregular operations and the computationally intensive parts can be implements efficiently.

IV. Thought

For the reason that the complicated computation part (CCP) which base on different domain-specific customization can be implemented by FPGA. But the irregular operations part (IOP) for design domain-specific customization cost large chip area. The whole system shows as Figure (10).

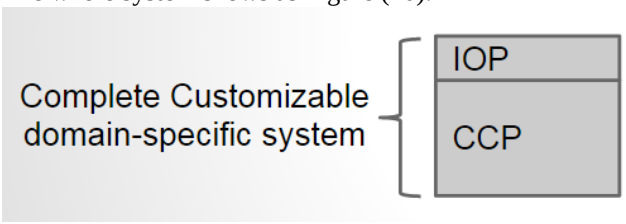

Figure (10)

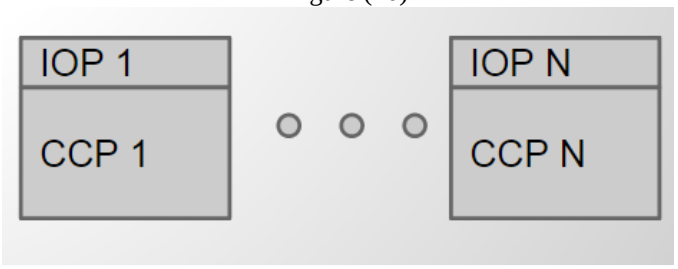

Figure (11)

To build the fully system, developer may combine multiple different domain-specific customization are design in the system. As the figure, the IOP are not been reused. It may cost a very large area. As the Figure (11), each design has its own IOP. It is not efficiently way to work on this.

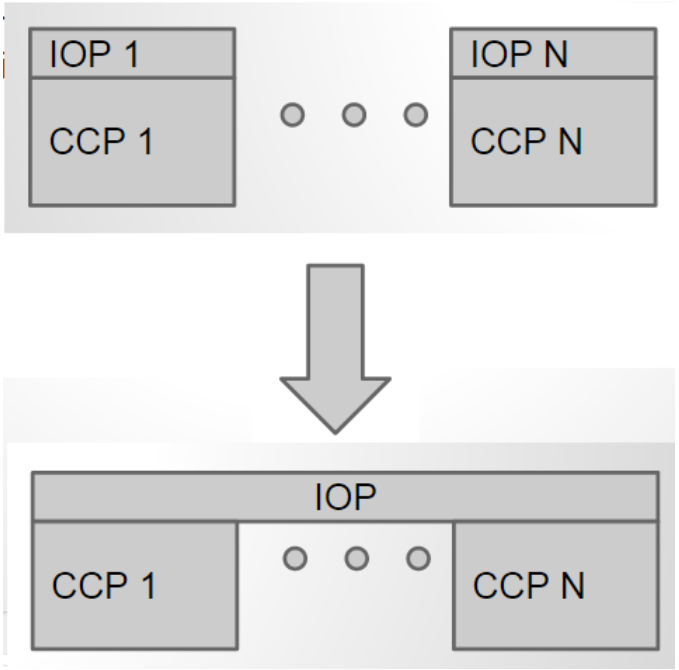

Figure (12)

However, the irregular operations part (IOP) is best match to the microprocessor architecture. Multiple different IOP customization design is hard to combination together. Therefore, design the microprocessors architecture that match IOP can get more efficiently. And share the same IOP for the

CCPs reduce significant area of the system. As the Figure (12), the IOPs are use same IOP resources. In this way, the area can be significant smaller than the origin one.

V. Design Purpose of my work

In the previous talking. The irregular operations parts of the customizable domain-specific computing system design plays an important role. As the analysis, the IOP can be significant decrease the area based on how we design the system. In the paper, the IOP design is best match to the microprocessor architecture. For this reason, we try to design or find an Instruction Set Architecture that suit for the IOP.

There is an Instruction Set Architecture called RISC-V. RISC-V is an open source implementation of a reduced instruction set computing (RISC) based instruction set architecture (ISA). RISC-V declare that this ISA is significant because it is designed to be useful in modern computerized devices such as warehouse-scale cloud computers, high-end mobile phones and the smallest embedded systems [34]. This ISA has a substantial body of supporting software, which fixes the usual weakness of new instruction sets [35]. RISC-V is also designed to be extensible from a 32-bit bare bones integer core suitable for a small embedded processor to 64 or 128-bit super and cloud computers with standard and special purpose extensions.

VI. Relate work of Risc-V ISA

RISC-V has 32 integer registers and 32 floating-point registers [36]. The memory is addressed by 8-bit bytes. The instructions must be aligned to 32-bit addresses [36]. Like many RISC designs, it is a "load-store" machine. The only instructions that access main memory are loads and stores. All arithmetic and logic operations occur between registers. The instruction set includes other features to increase a computer's speed, while reducing its cost and power usage. These include placing most-significant bits at a fixed location to speed sign-extension, and a bit-arrangement designed to reduce the number of multiplexers in a CPU. The designers claim that RISC-V CPUs achieve higher speeds, lower power and smaller, less-expensive electronics than some comparable commercial CPUs [37].

The basic 32-bit instruction set compiles to code that is larger than that of many other instruction sets. To compensate, RISC-V is designed as a variable-length instruction set [36][38]. The smaller, 16-bit subset is called RISC-V Compressed. A prototype includes 33 of the most frequently-used instructions, recoded into a more-compact 16-bit format. Research with the prototype showed that the code was 15% smaller than an x86 PC, roughly equivalent to MIPS' compressed code , and 7% larger than ARM's Thumb-2 code.

RISC-V also lacks condition codes, and even lacks a carry bit. The designers claim that this can simplify CPU designs by minimizing interactions between instructions [36][38]. On the contrary, RISC-V builds comparison operations into its conditional-jumps. Using the comparisons may slightly increase its power usage in some applications.

In many RISC ISA design, including a branch delay slot, a position after a branch instruction that can be filled with an instruction which is executed regardless of whether the branch is taken or not [36][38]. This feature can improve performance of pipelined processors, but it was omitted because it complicates in superscalar CPUs.

The RISC-V ISA was designed for research, and therefore includes extra space for new instructions [36]. The ISA include system instructions, atomic access, integer multiplication, floating-point arithmetic, bit-manipulation, decimal floating-point, multimedia and vector processing [36]. It also includes instructions for 32-bit, 64-bit and 128-bit integer and floating-point and was designed for 32-bit, 64-bit and 128-bit memory systems, with 32-bit models designed for lower power, 64-bit for higher performance, and 128-bit for future requirements. In this way, the ISA can operate with hypervisors, supporting virtualization. It was designed to conform to recent floating-point standards.

The RISC-V ISA define a special set of integer multiplication instructions, including a recommended sequence of instructions that a compiler can generate and a CPU can interpret to perform a fused multiply-accumulate operation.

VII. Instruction Set design of My work

Reference to the RISC-V instruction set architecture design, it is a powerful design for the Irregular Operation Part (IOP) design. But there is still a little weakness. Some of the instruction are no need for Complicated Computation Part (CCP). For example, if the CCP only compute in integer, no need of floating-point. It will cost more area in the useless function. For this reason, we redesign the Instruction set architecture and remove useless function. After Reduce and modify the Instruction set architecture. We called this Instruction set CPUKH

In CPUKH design, the 32-bit instruction set is the basic design. CPUKH has 16 register and total 31 basic instruction. This ISA significantly decrease the area and reserve a big space for extendable instruction set.

VIII. Hardware Pipeline Design of My work

Based on the CPUKH ISA, we design a RISC like pipeline architecture for hardware. This architecture is reference to the MIPS design. This pipeline architecture is design in five stage. It can has high speed and small area. Similar to the MIPS design [39][40][41], but creation parts is that we design a specific parts for extendable instruction, such as the figure (13).

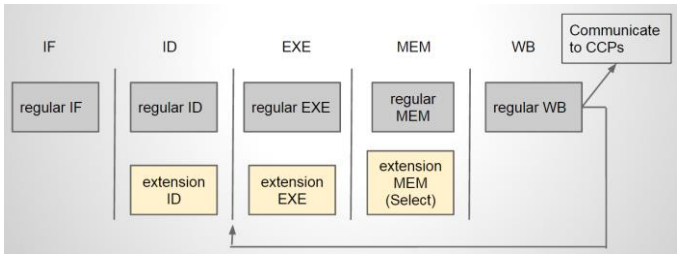

Figure (13)

There are three specific parts design for extendable instruction. In Instruction Decoder (ID), there is a special parts called "Extension ID". This part is design for new extendable instructions decode. Separate the basic instruction from the extendable instructions.

In execute stage (EXE), we design a special interface and specific block for add new functional computing for extendable instruction customizable computing function. If the customizable specific computing algorithm hardware is not that large. The computing hardware can be set in here. Otherwise, the large customizable specific computing algorithm hardware should be set in Complicated Computation Part (CCP). And using Communicate block in Write Back (WB) to communicate with the CCP that the CCP can be control and computing.

In the Memory stage (MEM), this stage is "Load and Store" stage of the memory. In this CPUKH architecture, the MEM stage only work when the instruction "Load" or "Store" happened. The extension MEM stage is used to load the data that the extension EXE stage needed. And also can store the data that come from the extension EXE stage computing data.

The Irregular Operation Part (IOP) has a communication network bus to control or connect the Complicated Computation Part (CCP). In CPUKH ISA design, the connection bus part is set in WB stage. After write back the data to the register file or some control signal extension registers, the CPU can get the overall data.

IX. Verify Environment of My work

The CPUKH hardware implement in Verilog code and verify on Xilinx Zedboard as the Figure (14) [42][43][44]. The Zedboard is a powerful platform SoC system that support FPGA testing environment. The Zedboard include Dual-core ARM processors, 512 MB DDR3, using AMBA for communication connection. This standard allow designer design an IP core for which system that adopt AMBA standard without any change. Helps designer quickly design a reusable circuit system. Thus, we design the CPUKH using AMBA standard. As done all the implementation and verify the whole design, we can quickly transplant the IP core to the other platform and system without any modify. Make the whole design flow more efficiently and decrease probability of error.

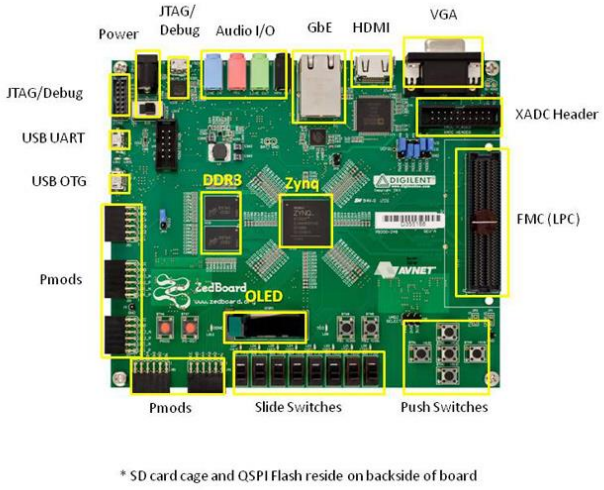

Figure (14)

In this paper, we implement a pure CPUKH without any Instruction extension. The pure core has general purpose computing capability. It can support most of the computing work and technical controlling.

We evaluate and test the whole system as the figure (15) [44]. The testing program are writing and programming in ARM CPU. "My Core" is the CPUKH ISA's implementation IP core "My Core" is seems like Irregular Operation Part (IOP). It connects to the ARM using AMBA standard and the other IP core such as LED and Switch is the testing hardware implementation like Complicated Computation Part (CCP). The Uart Console Monitor is the way to check the signal or the result is right or not. In this simple evaluating and testing system we can easily testing if the core is work or not.

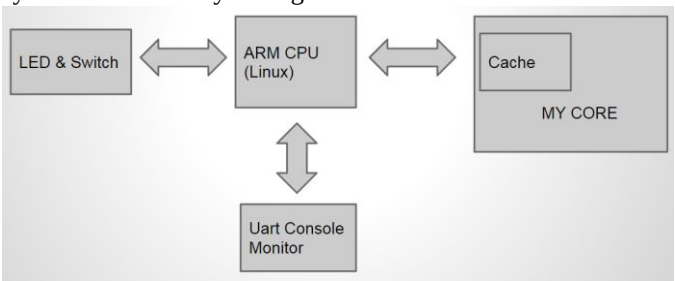

Figure (15)

We using Xilinx Platform Studio as tool to synthesis the circuit system [45][46][47]. As the result, we get the table as figure (16). This core and the bus interface consume 1671 registers and 2190 LUTs. The CPUKH system design implementation proportion of the total system is very small. Such as the table, the utilization shows the Slice registers cost only 1%. And the cost of the LUTs only have 4%. This mean the core is really tiny and can be used for Irregular Operation Part (IOP) in the customizable domain-specific computing system.

Slice Logic Utilization (whole system)	Used	Availabl e	Utilization
Number of Slice Registers	1671	106400	1%
Number used as Flip Flops	1670		
Number of Slice LUTs	2190	53200	4%
Number used as Logic	2157	53200	4%
Number of RAMB36	2	1827	1%
Number used as Memory	21	17400	1%

Figure (16)

In the figure (17), this table shows only the CPUKH core's cost without other IP core and Bus routing. The CPUKH design used 1406 Flip Flop and 1955 LUT, and the Maximum frequency of the core run in 73.362MHz. The result shows that this core has very tiny area cost and it has high speed performance. It means this core design may useful in Irregular Operation Part (IOP) in the domain-specific customization as the previous suppose.

Only Core	Flip Flops Used	LUTs Used
CPUKH Design	1406	1955

Figure (17)

X. COMPARE WITH CCP DESIGN

In the previous analysis, the design of customizable domain-specific computing system can be separate into two parts, IOP and CCP. The CCP is the specific block for the computationally intensive parts such as FFT, image processing…etc. Next, we will discuss the FFT area cost and compare to our design of the IOP. Comparing the two different parts that IOP may be the choosing option for domain-specific customization to make the work more efficiently and reduce the area cost.

We choose a FFT design as CPP to compare if the IOP can get the benefit or not. In the paper [48], this paper implement multi-FFT/IFFT core. The hardware acceleration is achieved by off laying transformation job onto uniformly connected FFT and IFFT core on an FPGA [48]. This design uses minimum gate resource and can accommodate n number of cores subjected to maximum available resource.

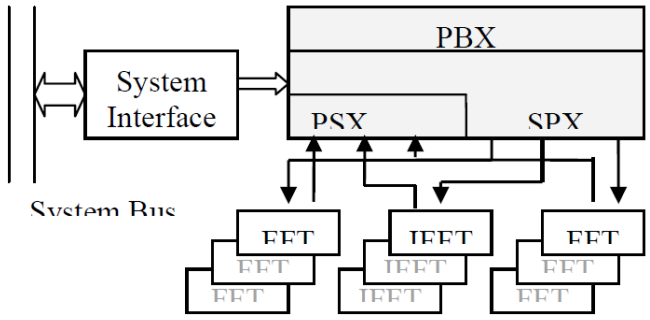

Figure (18)

In the figure (18), shows the system architecture design of FFT IP core [48]. The block representation of system to processing element crossbar path (SPX) & processing element to system crossbar path (PSX). This module takes care of the inward and outward flow of data from the system interface to the processing cores [48].

The SPX unit consists of two functional blocks as the figure (19) [48].

1. A 128x5 array for temporary storage of data termed as swap in here
2. Control logic to issue all control signals.

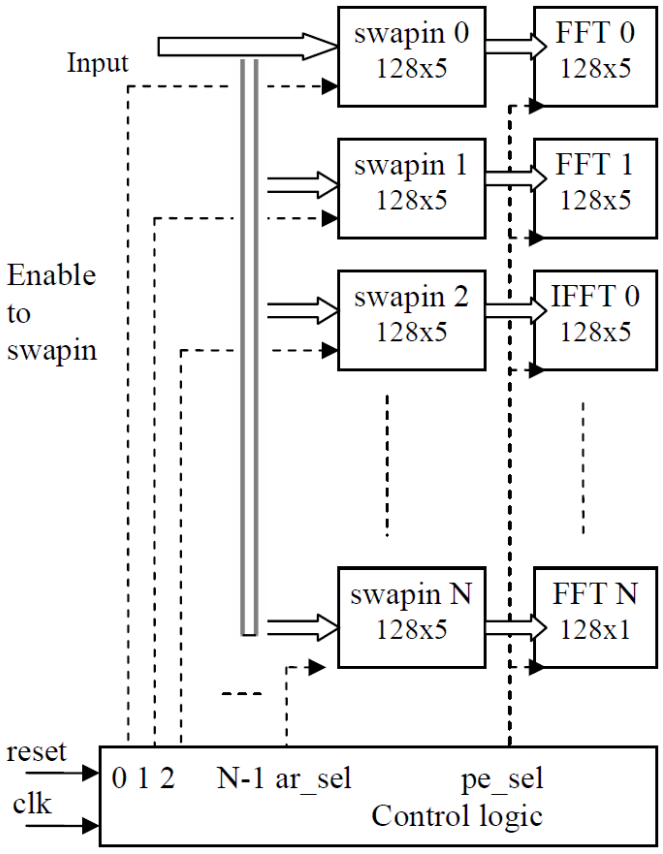

Figure (19)

The PSX unit also consists of three functional blocks as inscribed in Figure (20) [48].

1. 16x16 array to hold the output data from PEs termed as swap out here.
2. A 2:1 bus multiplexer to consolidate the data to 32 bit output.
3. Control logic to issue all control signals.

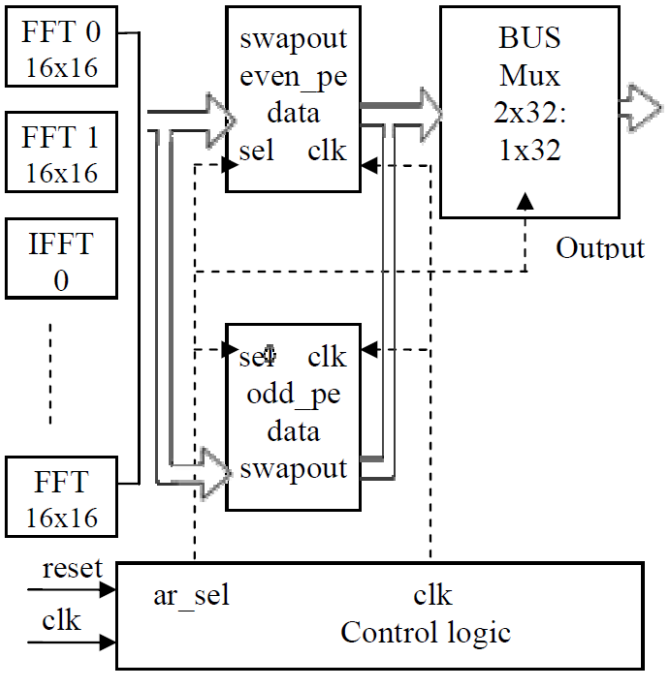

Figure (20)

As the origin analysis, the customizable domain-specific computing system can be separate into two parts, Irregular Operation Parts (IOP) and Complicated Computation Parts (CCP). As this FFT/IFFT system architecture design, the SPX's and PSX's control block can be regard as IOP. And the other block Array block can be regard as CCP. The IOP may match to the CPUKH architecture design.

The synthesis result of the complete system has been shown in the paper [48]. The 1 FFT/IFFT core cost 7125 slice registers and 15416 slice LUTs. The maximum frequency is 40.791MHz.

Compare to the CPUKH design, the CPUKH costs 1671 slice registers and 2190 slice LUTs. The CPUKH is significant smaller than the FFT/IFFT design. When the FFT/IFFT cores grows up. The 4 FFT/IFFT cores cost 27869 slice registers and 59752 slice LUTs. The maximum frequency is 42.602MHz. The 8 FFT/IFFT cores cost 54470 slice registers and 112476 slice LUTs. The maximum frequency is 38.124MHz. Compare to the CPUKH design, the cost of the CPUKH architecture is significant small than 1/10. If the FFT/IFFT cores IOP can be replace by CPUKH design. It may reduce the area significant. It may also efficiently and flexible for designing the whole system by using CPUKH architecture.

XI. Conclusion

In the figure (21), shows the complete Customizable domain-specific system architecture design. The Complicated Computation Part (CCP) such as FFT/IFFT core design can be separated into two part, Irregular Operation Parts and Complicate Computing Parts. This paper prove that the complete customizable domain-specific system architecture should include these two parts. As the analysis, the Irregular Operation Parts (IOP) match to the microprocessor architecture such as CPUKH. The IOP of the domain-specific customization system can be replace by CPUKH. Using the CPUKH as IOP can get benefit because of the reusable function in hardware. And the implementation of the design can be more efficient and flexible.

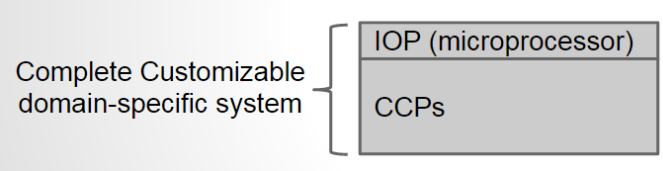

Figure (21)

XII. Contribute

In this paper, we analysis that Customization can be separate into IOP and CCP. We design and implement a microprocessor called "CPUKH" which are tiny, useful, general purpose, customizable, tunable and extendable. Next, we prove CCP and IOP are work in a real complete Customizable domain-specific system. Last, we prove the cost of our CPUKH design may get benefit from the complete Customizable domain-specific system design.

References

[1] J. M. Rabaey. Digital Integrated Circuits: A Design Perspec-tive. Prentice-Hall, Englewood Cliffs, NJ, 1995. USA: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx*.

[2] F. Blaabjerg, Z. Chen, and S. Kjaer, "Power electronics as efficient inter-face in dispersed power generation systems" IEEE Trans. Power Elec-tron., vol. 19, no. 5, pp. 1184-1194, Sep. 2004.W.-K. Chen, *Linear Networks and Systems*. Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] J. Cong, V. Sarkar, G. Reinman, and A. Bui, "Customizable Domain-Specific Computing", IEEE Design and Test of Computers, vol. 28, no. 2,pp. 6-201315, Mar. 2011

[4] U. Hoelzle and L. A. Barroso. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Morgan and Claypool Publishers, 2009.

[5] A. Ledeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom, and J. Sprinkle. Composing Domain-Specific Design Environments. Computer pages 44-201351, 2001.

[6] S. Che, J. Li, J.W. Sheaffer, K. Skadron, and J. Lach, "Accelerating Compute Intensive Applications with GPUs and

FPGAs", Proc. Symp. Application Specific Processors, pp. 101-107, 2008.
[7] B. Cope, P. Y. Cheung, W. Luk, and L. Howes, "Performance comparison of graphics processors to reconfigurable logic: A case study", IEEE Trans. Comput., vol. 59, no. 4, pp. 433-2013448, Apr.2010
[8] Estrin, G (2002). "Reconfigurable computer origins: the UCLA fixed-plus-variable (F+V) structure computer". IEEE Ann. Hist. Comput 24 (4): 3–9.
[9] Estrin, G., "Organization of Computer Systems—The Fixed Plus Variable Structure Computer," Proc. Western Joint Computer Conf., Western Joint Computer Conference, New York, 1960, pp. 33–40.
[10] C. Bobda: Introduction to Reconfigurable Computing: Architectures; Springer, 2007
[11] Hauser, John R. and Wawrzynek, John, "Garp: A MIPS Processor with a Reconfigurable Coprocessor," Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '97, April 16–18, 1997)
[12] History of FPGAs at the Wayback Machine (archived April 12, 2007)
[13] Andres, Kent (October 1970). A Texas Instruments Application Report: MOS programmable logic arrays. Texas Instruments. Bulletin CA-158. Report introduces the TMS2000 and TMS2200 series of mask programmable PLAs.
[14] Birkner, John (August 16, 1978). "Reduce random-logic complexity". Electronic Design (Rochelle, NJ: Hayden Publishing) 26 (17): 98–105.
[15] [Online] http://en.wikipedia.org/wiki/Programmable_logic_device
[16] "CPLD". xilinx.com. Retrieved 2013-11-17.
[17] "Complex Programmable Logic Device". blogspot.com. May 2008. Retrieved 2013-11-17.
[18] Wisniewski, Remigiusz (2009). Synthesis of compositional microprogram control units for programmable devices. Zielona Góra: University of Zielona Góra. p. 153. ISBN 978-83-7481-293-1.
[19] History of FPGAs at the Wayback Machine (archived April 12, 2007)
[20] Hauck, S., et al., "Totem: Domain-Specific Reconfigurable Logic," pp. 1-25, no date.
[21] S. Hauk and A. DeHon, Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation, Morgan Kaufmann, 2007.
[22] Jonathan Rose, et al., "Architecture of Field-Programmable Gate Arrays," Proceedings of the IEEE, Jul. 1993, pp. 1013-1041.
[23] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," in Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays, pp. 3–12, ACM Press, 2000.
[24] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 3, pp. 288–298, March 2004.
[25] V. Betz and J. Rose, "Circuit design, transistor sizing and wire layout of FPGA interconnect," in Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 171–174, 1999.
[26] H.-C. Hsieh, W. S. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, L. Tinkey, and R. Kanazawa, "Third-generation architecture boosts speed and density of field-programmable gate arrays," in Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 2/1–31.2/7, May 1990.
[27] H. Hsieh, et al., "Third-generation architecture boosts speed and density of field-programmable gate arrays," in Proc. 1990 CICC, pp. 31.2.1-31.2.7, May 1990.
[28] H. Hsieh, et al, "A 9000-gate user-programmable gate array, "in Proc. 1988 CICC, pp. 15.3.1-15.3.7, May 1988
[29] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal switch-module design for symmetric-array-based FPGAs," in Proceedings of the 1996 ACM Fourth International Symposium on Field-Programmable Gate Arrays, pp. 80–86, February 1996.
[30] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal switch modules for FPGA design," ACM Transactions Design Automation Electronic Systems, vol. 1, no. 1, pp. 80–101, 1996.
[31] H. Fan, J. Liu, Y.-L. Wu, and C.-C. Cheung, "On optimal hyperuniversal and rearrangeable switch box designs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 12, pp. 1637–1649, 2003.
[32] A. Shoa, S. Shirani, "Run-time reconfigurable systems for digital signal processing applications: a survey", The Journal of VLSI Signal Processing, 39 (March 2005), pp. 213–235
[33] K. Bondalapati and V. K. Prasanna "Reconfigurable computing systems", Proc. IEEE, vol. 90, no. 7, pp.1201 -1217 2002
[34] [Online] http://en.wikipedia.org/wiki/RISC-V
[35] Waterman, Andrew; Lee, Yunsup; Patterson, David A.; Asanovi, Krste. "The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA version 2 (Technical Report EECS-2014-54)". University of California, Berkeley. Retrieved December 26, 2014.
[36] Waterman, Andrew, et. al. "The RISC-V Instruction Set Manual Vol. I, User-Level ISA, version 2.0". RISC-V Downloads. Regents of the University of California. Retrieved 2014-08-25.
[37] "Rocket Core Generator". RISC-V. Regents of the University of California. Retrieved 1 October 2014.
[38] Waterman, Andrew (May 13, 2011). Improving Energy Efficiency and Reducing Code Size with RISC-V Compressed. U.C. Berkeley: Regents of the University of California. p. 32. Retrieved 2014-08-25.
[39] "MIPS32 Architecture". Imagination Technologies. Retrieved 4 Jan 2014.
[40] "MIPS64 Architecture". Imagination Technologies. Retrieved 4 Jan 2014.
[41] Rubio, Victor P. "A FPGA Implementation of a MIPS RISC Processor for Computer Architecture Education" (PDF). New Mexico State University. Retrieved 22 December 2011.
[42] Cadence: "Virtual Platform for Xilinx Zynq-7000 EPP User Guide", Product Ver. 11. 10. 055, Apr. 2012
[43] Avnet, Inc. : "ZedBoard" http://www. zedboard. Org
[44] P. Wehner, M. Ferger, D. Go¿¿hringer, M. Hu¿¿bner: "Rapid Prototyping of a Portable HW/SW Co-Design on the Virtual Zynq Platform using SystemC", In Proc. of the 26th IEEE International Systems-on-Chip Conference (SOCC), Erlangen, Germany, Sept. 2013.
[45] V. Asokan, "Designing Multiprocessor Systems in Platform Studio," XPS White Pajier, WP262, 2007.
[46] William Wong, "Basics of Embedded Processor Design," Supplement to Electronic Design magazine, February 16, 2006
[47] Vasanth Asokan, "Dual Processor Reference Design Suite," Xilinx application note, APP996, 2007.
[48] Hassan Raza, "High throughput design & implementation of multi-FFT/IFFT core in FPGA for hardware acceleration", Department of Electronics & Computer Sc. Engineering, Visvesvaraya National Institute of Technology, Nagpur – 440010, INDIA

Author Kun-Hua Huang

Was born in Taiwan, in 1991. Interesting in Computer Science. Expert in microprocessor system design. I have built a 16-bit microprocessor on the FPGA board. I design the ISA (Instruction set architecture) and design five stage pipeline of the microprocessor architecture. Then, using Verilog code to implement the microprocessor architecture on the Xilinx FPGA board XC3S2000. Also, I design the whole system for this embedded system that make the microprocessor work and used to compute and control with the other component (Like LED, Monitor, PS/2 keyboard, Ultrasonic sensor...).

And redesign a new 32-bit microprocessor architecture for general purpose computing. The FPGA designer can use the core in their system design easier and flexible.

Not only play in the microprocessor design and implementation in hardware, but also learn in the software. Learned the LLVM compiler. Also learn in embedded system, graphic. Contribute the code to u-boot.