White Paper

A Multi-FPGA Based Platform for emulating a 100M-transistor-scale Processor with High-speed Peripherals

2009

This published paper was solely written by Institute of Computing Technology (ICT), Chinese Academy of Science. It describes how ICT used S2C Dual Virtex-5 TAI Logic Modules to prototype a 100 million transistor-scale processor at 25MHz to boot unmodified operating system for carrying out a variety of architectural explorations. The paper identified several key challenges when prototyping a complex design onto multiple FPGA devices and how the ICT research engineers were able to solve these challenges including FPGA partitioning, pin limitations, emulating high-speed IO and debugging the design on S2C's TAI Logic Module.

S2C Inc.
1754 Technology Drive, Suite 620
San Jose, CA 95110, USA
Tel: +1 408 213 8818
Fax: +1 408 549 9948
www.s2cinc.com

A Multi-FPGA Based Platform for emulating a 100M-transistor-scale Processor with High-speed Peripherals

Weiwu Hu[1], Huandong Wang[1,2], Xiang Gao[1], Yunji Chen[1], Dan Tang[1,2]
[1]*Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*
[2]*Graduate University of Chinese Academy of Sciences, Beijing 100049, China*
{hww, wanghuandong, gaoxiang, cyj, tangdan, }@ict.ac.cn

ABSTRACT

Software simulation has become a powerful method in development of processors, but to simulate processors with a state of art architecture has become extremely difficult. The simulation of this kind of the processors is time-costing as the scale becomes larger and larger complied with the Moore's low. Accompanied with the enlargement of scale, many high-speed IOs are integrated into processors as SOC chips, causing another problem that the verification of high speed IOs is very limited for software simulation. With the help of FPGA based platform, the verification and evaluation of processors can be done in a relative high speed. The FPGA based platform also provides a real environment with a lot of real chips working together for the verification of high speed IOs. In this work, a multi-FPGA based platform used for verification and evaluation of Godson-2G processor is introduced. Tactics for a semi-custom partitioning and the design flow are discussed. Method to emulate DDR and HyperTransport PHYs is offered after that. Instrumentation based debugging and the performance evaluations are both presented at the end of this paper.

Categories and Subject Descriptors

C.1.0 [Processor Architectures]: General

General Terms

Design, Verification, Performance

Keywords

Godson-2G, Multi-FPGA, emulate, verification, evaluation

1. INTRODUCTION

Today's general-purpose processors have been evolving towards having more functional peripherals on-chip to provide higher performance and better cost-effective solutions. AMD has been using on-chip memory controller and HyperTransport [1] [2] links for years in its Athlon64 and Opteron [3] processors. Intel's latest Nehalem [4] processor family also departs from the traditional front side bus and integrates the memory controller into the CPU. Furthermore, other system components such as PCI-E, GPU, finding their positions in general-purpose processors has been a matter of "when", rather than "if".

With the unprecedented level of chip integration leading to much more design and verification efforts, processor designers are facing great challenges. More peripheral components added into the chip expand the already huge design space. Interactions between components get much more complicated. System-level evaluation and decision-making are suffocated by the slow simulation speeds. Meanwhile, the lack of detailed and ample peripheral modeling in the RTL-level simulation environments brings high risk into the pre-silicon verification.

To address challenges from design exploration and pre-silicon verification, more and more processor designer resort to FPGA-based prototype. As widely-deployed platforms in the area of SoCs and ASICs, the flexibility, speed, and enormous capacity qualifies them for the emulation of large and complex systems. Nevertheless, a typical state of art processor always exceeds the capacity of the largest modern FPGA device. The largest FPGAs before 2009 in production have an estimated equivalent capacity of about 1.5 million gates or 30 million transistors. Prototyping a modern processor larger than this means spreading the functionality across multiple FPGAs. It is a complicated task, but one that is well worth the effort.

This paper presents a multi-FPGA based platform (called MFP) which prototype the Godson-2G (also known as Loongson-2G), a modern superscalar processor with transistor number exceeding 100M .To the best of our knowledge, the MFP is the first platform capable of emulating a state of art processors across multi-FPGA. Taking into account the trend toward ever-increasing levels of processor integration, we view Multi-FPGA as inevitable in the long term for the pre-silicon functional verification and architectural evaluation. It's also a very promising and straightforward step to emulate huge ASICs/SOCs chips.

In the MFP system, different kinds of I/O peripheral are working together. The emulated processor core can run at up to 25 MHz. As well, the integrated DDR2/3 controller can work at 40 MHz and HyperTransport controller at 25 MHz. At such speeds, it is sufficient to boot unmodified operating system, enabling processor designer to carry out a diverse variety of architectural explorations. Time-consuming benchmarks, including both computing-intensive and IO-intensive applications, can be completely and fast evaluated. For example, the entire SPEC CPU 2K benchmark in train scale can be completed within twelve hours, in contrast to about a week by the Xtreme emulation [5], resulting in a significant improvement into the pre-silicon performance prediction.

The key contributions of the work include:

- We analyze a 100M-transistor-scale superscalar processor implementation on two virtex-5 FPGAs. Methodology to partition, synthesis, and map an almost-unmodified RTL design to a multi-FGPA target are detailed.
- We present the experiences under FPGA environment to emulate the high-speed I/O peripherals, including Hyper-Transport and DDR2/3.
- System-level Debugging, we share our methods and experience in tackling the problem at system-level developing.
- Preliminary evaluation of pre-silicon performance, including computing-intensive and I/O-intensive bench-marks validation is presented.

The remaining sections of this paper are organized as following. Section 2 summarizes related work and introduces the architecture of Godson-2G micro-processor. Section 3 describes the multiple FPGA platform used in this work. Section 4 discusses the semi-custom partitioning and design flow. Section 5 presents the method to emulate the high speed IO. Section 6 details a debugging method in FPGA and Section 7 gives some evaluation results.

2. BACKGROUND

As the scale of chip design increasing rapidly, it is difficult to evaluate a design only by software simulation. Although the emergency of multi-FPGA based [6] or hardware based emulation accelerator greatly speeds up simulation or emulation, IO behaviors and different IO specifications are still hard to verify. While the capability of a single FPGA platform is too small for emulating a 100M-transistor-scale processor, a multi-FPGA based platform is one of the best way to verify and evaluate the design of Godson-2G micro-processor.

2.1. Related Work

The three main options for evaluation and verification open to processor designers are simulation, emulation, and FPGA-based prototypes. The ever-increasing chip scale makes the software simulation notoriously slow to simulate an entire chip at RTL-accurate level. The fastest true RTL-accurate simulators of modern processors run at about 1 KHz to 10 KHz, which means two minutes of simulated time in approximately corresponding to one to ten years of simulation time. At such speeds, it is impractical to use realistic program to explore, evaluate and refine micro-architectures.

Hardware-based emulation is another reasonably popular technique, which can be viewed as a kind of accelerated and board-pluggable simulation. Several companies such as Cadence/(Xtreme,Palladium [7]), Mentor, Synopsys, EVE and Tharas sell FPGA-based accelerators, emulators or tools that take arbitrary RTL and map it to hardware to improve simulation performance. The emulators tend to be very expensive due to their complexity. With an equivalent speed of only 500 KHz to around 2MHz, however, it's still two to three orders slower than the real chip. As peripheral I/O devices in the emulation system are working at their real speed, the extreme speed imbalance between I/O peripheral and the emulated processor will introduce many odd obstacles to system-level evaluations. Taking the TCP/IP connection as an example, low frequency of the processor lead to unexpected packet losses and connection termination. Besides, I/O Bus adhere to the emulated processor should also work at a much lower speed than the sane frequency, causing I/O devices on the bus behave abnormally.

For almost all mainstream architectures, there have been FPGA-synthesizable processor designs, including ARM [8], MIPS [9], SPARC [10], Itanium [11], and PowerPC [12]. Lu et al. [13] previously presented an FPGA-synthesizable version of Intel's vintage Pentium processor. Another notable work was the Intel Atom prototyping reported in [14], which for the first time disclosed Intel's effort to implement an FPGA-synthesizable version of a modern x86 processor.

But the modern processor can accommodate only a fraction of the whole processor chip, making system-level validation still a dilemma.

Fortunately, Moore's law has not only enabled these dense multi-core chips, it has also enabled extremely dense FPGAs.

Compared with powerful modern processor, processor cores mentioned above are relatively simple and small. Even the biggest one of them, namely Atom, consumes a total of 47.2 million transistors, allowing the whole design being mapped into a single FPGA. Contrary to these approaches, we implement the whole processor chip into multiple FPGAs, which removes the chip-size limitation caused by the bounded capacity of a single FPGA device. Furthermore, on-chip peripheral interfaces are also integrated, making the system-level validation feasible and reliable.

The RAMP collaboration [15] is building the necessary infrastructure to compose systems of up to 1024 processors. It is targeted at facilitating software research in many-core era. The processor cores adopted by RAMP are simple in complexity and small in size. One to two dozen cores can be programmed into a single FPGA. However, it lacks the ability to validate a state of art commodity processor that exceeds the capacity of a single FPGA.

2.2. The Godson-2G Processor

The Godson project, which was initiated by ICT-CAS in 2001, was the first attempt to develop high performance general purpose processors in China [16]. Godson-2G, the latest member of Godson processors supposed to tape out in 2009, is the single core version of Godson-3 [17]. The chip measures 53.90 mm^2 in size and consumes 106.8 million transistors. The estimated peak frequency is up to 1GHz and the power dissipation is 5-7 Watt depending on applications.

The architecture of Godson-2G is shown in Fig.1. The chip is based on two crossbars with 128-bit width data buses. The Crossbar with 2x2 ports called Level-1 Xbar connects the processor core, L2 Cache and HyperTransport controller. The other one with 2x3 ports called Level-2 Xbar slots in between L2 Caches, DDR2/3 controllers and PCI/LPC interface.

The processor core of Godson-2G (named GS464) implements MIPS64 instruction set [18] in a four-issue, out-of-order execution way [19]. It fetches and decodes four instructions per cycle and dynamically issues them to five fully pipelined functional units (two fix-point, two floating-point and one memory access unit). Instructions are issued out of order by two 16-entry reservation stations and are committed in program order by a 64-entry reorder queue. Two 64-entry physical register files are used for register renaming of the general purpose and floating point registers separately. An 8192-entry pattern history table, a 9-bit global history register, a 16-entry branch target buffer and a 4-entry return address stack keep the branch history information for prediction. GS464 has a 64KB instruction cache and a 64KB data cache. A 24-

entry memory access queue that contains a content-addressable memory for dynamic memory disambiguation supports out-of-order memory access and non-blocking cache in GS464.

The Godson-2G supports a broad range of IO interfaces. These peripherals can be classified into two categories: the high speed IO and the low speed IO. High-speed IO controllers are critical for efficient executions of throughput based workloads, while low speed IO controllers provide flexibility to form a functionally complete system. The current version of Godson-2G encompasses one high-speed HyperTransport controller. The low speed IO interfaces in Godson-2G include PCI/LPC controller, UART controller, SPI controller and GPIO controller.

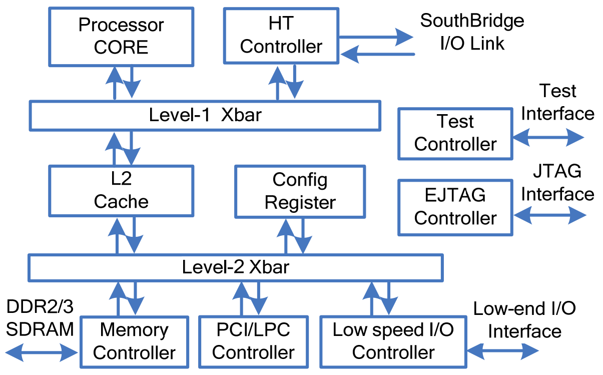

Figure 1. Architecture of Godson-2G

The two crossbars inside the Godson-2G processor are running different protocol. A cache coherence supported AXI protocol [20] is running on the Level-1 Xbar, while a standard AXI protocol is running on the Level-2 Xbar. Devices with requirement of cache coherence are placed onto the level-1 crossbar, such as L2 cache and Hyper-Transport controller. As the HyperTransport controller is placed on the Level-1 Xbar, cache coherence protocol between the native cache and IO device or remote cache in the other processors connected by the HyperTransport bus can be simply maintained. [21]

The low speed devices without cache coherence requirement are placed onto the level-2 crossbar, such as PCI/LPC controller, DDR2/3 SDRAM controller, UART controller, SPI controller, and GPIO controller. Without cache coherence protocol supported, software method can be used to maintain the coherence with caches. [22]

3. STRUCTURE OF THE PLATFORM

As a high performance SOC chip, pre-silicon validation in the FPGA platform of Godson-2G raise compelling requirements, including:

- Completeness. The capacity of the platform should be big enough to accommodate the whole chip design.
- Flexibility. The platform should be flexible for evaluating different kinds of application, including benchmarks for computing power and IO performance, running different kinds of peripheral devices.
- Applicability. Fast and accurate performance prediction for both IO-intensive and computing-intensive application.

To address these requirements and challenges, we propose a two-board solution to spread system functionality across two boards. The resulting MFP platform is constructed by a mother board and an FPGA daughter board. The mother board contains all necessary building peripheral chips and device slots, while the FPGA daughter board hosts the whole Godson-2G. To ease hardware platform setup, S2C's Dual Virtex-5 TAI Logic Module board is used as the FPGA daughter board. Two boards are erected together through high-speed PCB-to-PCB connectors. The photo of the Godson-2G MFP platform is shown in Fig. 2. The FPGA daughter board is placed above the mother board in the left.

3.1. Mother Board

The mother board contains the LPC interface part, the PCI interface part, SPI flash, UART ports and the south bridge part. LPC interface is connected to a LPC flash which stores the CPU booting codes, a nixie tube as debugging display and a super IO chip. The super IO chip is used for the function of serial port, PS/2 ports for keyboard and mouse. The PCI interface is led to two standard 32-bit PCI sockets. The LPC, SPI, UART and PCI interfaces on the mother board are directly connected to the LPC/PCI controller and the Low-speed controller in the FPGAs.

To evaluate the compatibility of the HyperTransport interface, two different mother boards are designed. The first one is using an Nvidia chipset as south bridge as shown in Fig.2, the other one is using AMD chipset, not shown here.

The chipsets in the mother boards are connected to the upper FPGA through HyperTransport interface. Most of the right half of two mother boards is occupied by various peripherals of the south bridge, including PATA socket, SATA sockets, PCI sockets, PCI-E sockets, VGA port, LPC interface and GMAC interface with a GMAC PHY chip on board.

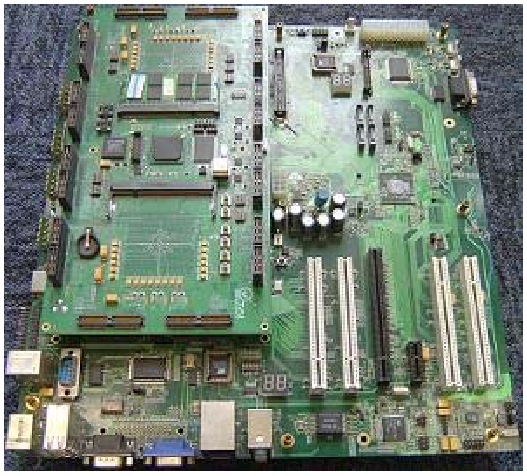

Figure 2. FPGA platform of Godson-2G

3.2. FPGA Daughter Board

The Virtex-5 330 [23] is a Xilinx FPGA with about 330K logic cells inside. The capacity of this FPGA is the largest before 2009, but still too small for this design. S2C Dual Virtex-5 330 TAI Logic Module is designed for rapid SoC/ASIC prototyping. One TAI Logic Module is equipped with two Virtex-5 330 FPGA (as shown in Fig.3, called Upper FPGA and Lower FPGA), and has two DDR2 SO-DIMM sockets for memory extensive applications. There are 600 wires connecting between the two FPGAs, and many other pins routed to twelve IO connectors distributed on the daughter boards.

Three of the IO connectors are used in this work. The one from Lower FPGA is used for LPC interfaces; two from the upper FPGA are used, one for HyperTransport link, the other one for LPC, PCI, SPI and UART interfaces.

Figure 3. Boards of FPGA platform

4. PARTITIONING AND SYNTHESIZING

In the whole design flow, partitioning is the most important and complicated phase; afterward both parts of the partitioned design are synthesized separately and mapped into two FPGA chips on the same daughter board. Although EDA vendors has provided software tools such as Certify [24] to build multi-FPGA based prototypes of ASIC design in an easy, intuitive fashion, special care must be taken to avoid an inefficient solution when applying to a state of art processor like Godson-2G .

An unevenly partitioned design can easily consume 100% of the I/O resources on a device while at the same time utilizing only a relatively small amount of its internal logic resources. In the MFP implementation, we develop a semi-custom design flow. The partitioning in MFP is manually done to take into consideration the interaction between components in different FPGAs. Another purpose of manual partition is to avoid multiplex groups of IOs together, which may result in a difficult, time-consuming and cumbersome development cycle.

As for synthesizing, the RTL code of Godson-2G is entirely written in strictly synthesizable verilog language.

4.1. Partitioning Metrics

With two FPGA chips involved, it is important to partition the whole design into two parts, synthesize separately and map into corresponding FPGA. The way to partition is mainly determined by three facts. The first is resource utilization of each chip, the second is number of wires left for communication between the two chips, and the third is the location of IO interface from the mother board.

Partition may be a complicated work for most multi-FPGA designs. The worst case is that times of iteration might take place when a partition can not fit well. To avoid this problem, we do a semi-custom partition in the view of architecture. We make some modification to the original architecture of Godson-2G and partition it into two subsystems. The first one is computing subsystem with only processor core and LPC controller embedded, with the purpose to be verified independently in a single FPGA system by fetching instructions and printing debugging messages through the LPC interface. The other one is an IO and memory subsystem including the rest parts of Godson-2G.

The architecture of computing subsystem just looks like the Lower FPGA in Fig.4. Some tailored application with usage of only LPC bus such as serial port, LPC flash and nixie tube can be run on this system. The width of bus used for interconnection between the two subsystems is no more than 600, making direct communication between two FPGA possible. Direct communication proves great convenience for a multi-FPGA design, as no frequency doubling is needed, causing no frequency degression.

With full acknowledgement that the Lower FPGA with only the processor core and PCI/LPC controller embedded works well, the problem left is just to deal with the upper FPGA, which contains the IO and memory subsystem.

Considering all reasons mentioned above, architecture of Godson-2G FPGA is divided into two parts, as shown in Fig.4. Most of the IO interfaces are mapped into the Upper FPGA, with a replicate LPC controller integrated in the Lower FPGA marked as dotted lines in Fig.4, which can be bypassed when the two FPGAs working together.

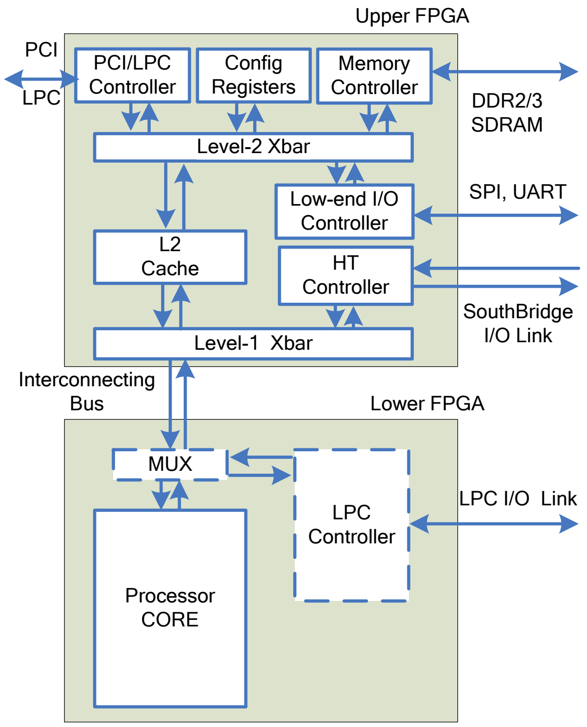

Figure 4. Architecture of Godson-2G FPGA

The interconnecting bus between two FPGA is the original processor core interface which contains no more than 600 wires. As a LPC or PCI access need not to be cached, the replicate LPC controller can be placed into the Lower FPGA and connected directly to the processor core as a slave device by a multiplexer which is enabled when debugging and disabled at normal mode. Placing the LPC controller into the Lower FPGA can make full utilization of the IO pins in Lower FPGA and also make the Lower FPGA can be verified separately and before the whole system ready but also brings a little difference from the original architecture of Godson-2G.

Thus the Lower FPGA contains a basic system which can boot up the processor core with BIOS codes stored in LPC flash and interactive through serial port by the LPC bus. The upper FPGA contains the IO subsystem, including L2 cache, memory controller and HyperTransport controller and all other interfaces of Godson-2G.

4.2. Design Flow

The entire design flow for verification and evaluation on Multi-FPGA platform is shown in Fig.5.

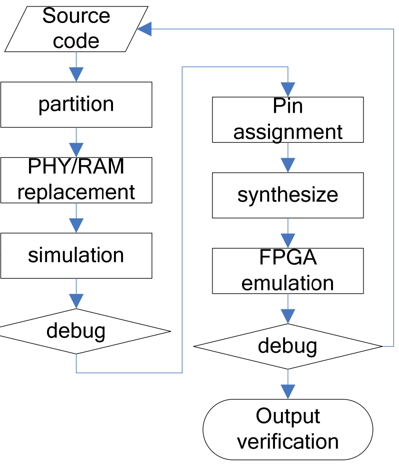

Figure 5. Flow of FPGA Verification

Original source code of Godson-2G is modified for the FPGA architecture and partitioned into Uint-1 and Unit-2 for the two pieces of FPGA. Some modules need not be verified in FPGA are removed, including the test controller, ejtag controller.

FPGA PHYs (SERDES) are used to replace the original ASIC PHYs in the design, including Hyper-Transport PHY and DDR2/3 PHY, which are hard cores in ASIC design. ASIC RAMs should be replaced by FPGA RAMs, too.

That is the all modification done to source code. Before synthesizing, source code of FPGA is first verified using simulation tools in random verification environment to avoid any low class bugs that may be injected by the modification.

All pins of the two FPGA are then allocated by their bus location and corresponding IO standard.

After all these have been done, source codes for Unit-1 and Unit-2 is then synthesized separately in ISE, a synthesizing tool for Xilinx FPGA, to generate the binary files for downloading.

A lot of applications including BIOS, Linux and different kinds of benchmarks are run and debugged on the MFP at the phase of FPGA emulation.

Resource utilizations of each FPGA are shown in Table 1.

Table 1. Resource utilization of each FPGA

	Upper FPGA	**Lower FPGA**
Logic Utilization		
Slice registers	46%	27%
Slice LUTs	65%	77%
Logic Distribution		
Occupied slices	79%	88%
IO Utilization		
Bounded IOBs	67%	43%
Block RAM	56%	78%

Utilization from both FPGA synthesizes and ASIC designs are shown in Table 2. There are some differences in the utilization between FPGA and ASIC. That many be caused by the different treatment about RAM and different cells used in FPGA and ASIC. Each slice of FPGA contains several registers and combining logic; while each instance of ASIC may be a register, a logic gate or a hard core, causing difference between the two designs.

Table 2. Resource utilization of FPGA and ASIC

	FPGA slice number	**Percent**	**ASIC instance number**	**Percent**
Processor core	93182	56%	800171	41%
HyperTransport	22041	13%	427651	22%
DDR2/3	19167	11%	405685	21%
Level-1 Xbar	13907	8%	68222	4%
Level-2 Xbar	8419	5%	56603	3%
Config registers	2102	1%	19987	1%
Level-2 cache	5883	4%	160124	8%

5. EMULATING HIGH-SPEED IO

Godson-2G is a high performance SOC with high-speed peripherals, including DDR2/3 SDRAM interface and HyperTransport interface.

One of the targets of Multi-FPGA platform is to verify the peripheral interfaces and their interactivities with the third-party chips. That's why we designed two different mother boards with different chipsets. The emulation of the DDR2/3 and HyperTransport PHY is an important step to achieve this goal. PHY (or SERDES) is a full custom designed hard core embedded on chip, used for high-speed transmission. The interface frequency of HyperTransport PHY in Godson-2G is 800MHz and DDR2/3 PHY is 400MHz. As for FPGA prototyping, this part must be re-implemented especially to emulate the function of the hard core in physical design.

The FPGA PHY of HyperTransport targets at 200 MHz and the DDR2/3 PHY targets at 40 MHz.

The main function of PHY is to convert between a set of parallel signals and a serial DDR (Double Data Rate) signal for on-board transmission. Driver of PHY is used to transform the internal parallel signals to external serial signal. Receiver of PHY is used to transform the external serial signal to internal parallel signals.

Another function is clock phase adjustment. Both output clock of driver and input clock of receiver might need to be adjusted according to the protocol.

5.1. HyperTransport Interface

The link frequency of HyperTransport PHY must be at least 200MHz due to HyperTransport specification. So that the corresponding control logic should run at 100MHz to communicate with south bridge.

As For the HyperTransport interface, ratio of parallel data to its corresponding serial signal is 4:1. For example, the waveform of driver signals is shown in Fig.6. The four parallel data from the control logic named PHY_Dn (n = 0...3) are driven by the clock named PHY_clk which is running at 100MHz. They are serialized to generate the link signal HT_cad. Clock frequency is doubled and clock phase is postponed according to the protocol requirement. For the receiver part of PHY, it just reverses the course to de-serialize the four PHY_Dn data from the external HT_cad and restores the half-frequency clock from the HT_clk.

With the implementation of FPGA PHY, frequency of HyperTransport interface is 200 MHz after system cold reset, with a link width of 8 bits. A set of asynchronous FIFO is placed between PHY and controller to transmit the data between 100 MHz clock domain of PHY and 25 MHz clock domain of the controller.

Placement of PHY can be viewed in the architecture of HyperTransport interface in Fig.7. The Tx control logic and the Rx control logic are running at 100 MHz, while the HT controller is running at 25 MHz with three sets of asynchronous FIFOs between them.

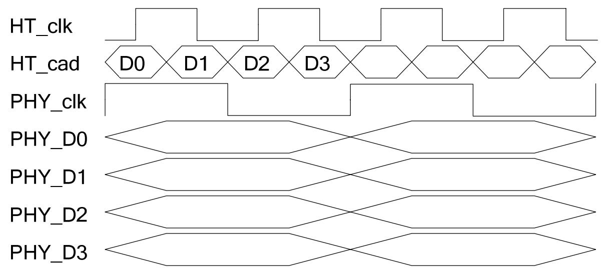

Figure 6. Waveform of HyperTransport PHY

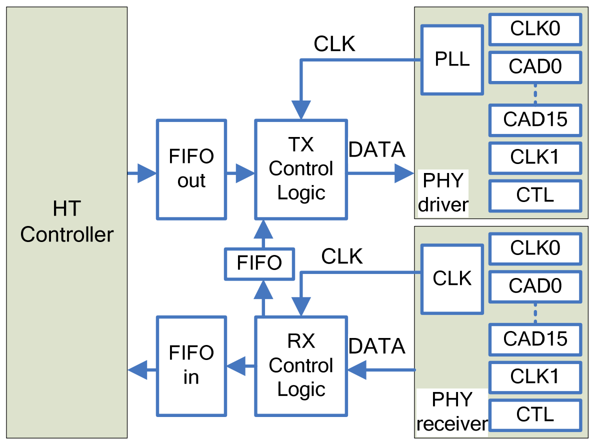

Figure 7. Architecture of HyperTransport interface

5.2. DDR2/3 Interface

Frequency requirement of DDR link is not as restrict as HyperTransport link. Thus the DDR2/3 PHY can also run at 40 MHz as DDR2/3 controller in the same clock domain.

The signal processing method of the DDR2/3 PHY is similar to HyperTransport PHY. Both them are double data rate compared to clock frequency.

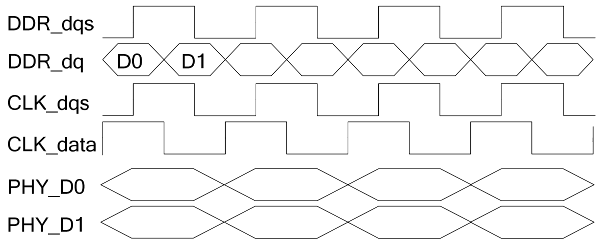

Figure 8. Waveform of DDR2/3 PHY

In order to centralize the DDR_dqs signal to the DDR_dq signals in a memory write access, two clocks are introduced into DDR2/3 PHY, as shown in Fig.8, marked as CLK_dqs and CLK_data. The latency from CLK_dqs which is responding for DDR_dqs to CLK_data which is responding for DDR_dq is one-forth clock phase.

For read dqs and read data, the DDR2/3 PHY is also responding for centralizing the positive edge of DDR_dqs to the DDR_dq signals, de-serializing and synchronizing the data to memory controller clock domain.

As there is no phase separation between read dqs and read data, read dqs must be delayed one-forth clock phase to sample and de-serialize the read data. Thus a set of delay line logic is placed into the DDR2/3 PHY for each read dqs.

6. INTEGRATION AND DEBUGGING

In MPF validation, the processor is tested in a system setting. Validation in this setting concentrates not only on the processor, but also interaction with chipset, memory system, and other peripherals.

6.1. System Environment

The configuration of processor core and peripherals are shown in Table 3. The caches are chopped to 512KB from 1MB of ASIC design to fit the capacity of FPGA.

Table 3. FPGA Configuration of Processor

	Characters	Frequency
Processor core	4 issues Out of order 64KB I-cache 64KB D-cache	25MHz
L2 cache	512 KB	25MHz
Memory controller	64 bit DDR2 1GB	40MHz
HyperTransport controller		25MHz
HyperTransport Bus	8 bit width / 16 bit width	200MHz
PCI/LPC controller		33MHz
L1/L2 Xbar	128 bit width	25MHz
Peripheral devices	10M/100M NIC on PCI GMAC on South bridge PATA on South bridge Flash, PS/2 on LPC Serial port on LPC UART port, SPI flash	

6.2. Software Stack

PMON will be first booted up on the Multi-FPGA platform after system reset. It will first initialize the processor core, caches, and then the memory controller. After that, it will initialize the HyperTransport interface and the devices on south bridge.

After everything is initialized, a network card on the PCI bus of south bridge is activated. Linux kernel is downloaded by the Ethernet link from a tftp sever.

When Linux is running, many other devices on south bridge are set up to work, including the IDE hard disk, GMAC Ethernet controller, and etc.

Many things can be done on Linux operating system. A lot of benchmarks are tested, including SPEC CPU 2000, LMbench, ftp downloading, stream test, and etc.

Some of the testing results are given in the following section.

6.3. Debugging

The best place to debug a processor bug is in the simulation environment, where all signals are available for scrutiny. Unfortunately, in the system environment of real system almost no internal signals are visible to the system debugger. A suite of tools was developed for use in the Pentium processor [25], using both architectural and micro-architectural features of the processor.

FPGA platform is also hard to debug when any abnormality emerges, as signals inside are difficult to probe. But bugs are not occasional at the very beginning. Thus the way to debug must be under consideration.

Some observing points are injected into the DMA path from HyperTransport controller to the memory controller for checking where a data corruption has happened.

Each observing point is a monitor that can be triggered by a software programmable pattern. All of the monitors are chained together in a daisy-chain topology by two sets of buses called configure bus and configure return, which are controlled by the configuration registers module as Fig.9.

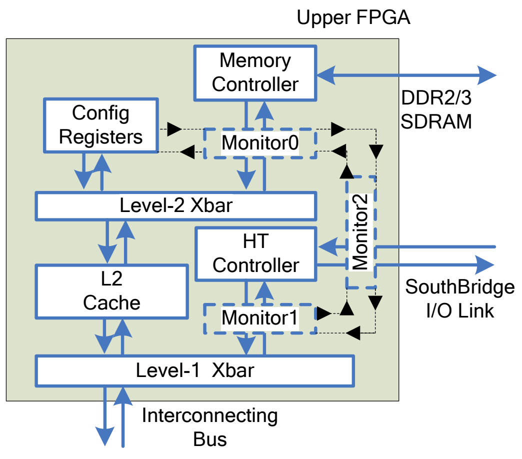

Figure 9. Injection of Monitors

The configure bus transmits the read or write command with address and write data, while configure return contains the response from the selected monitor. Software can access the monitors by accessing the configuration registers module.

With the help of injected monitors, it is easy to tell where the problem may be.

We once encountered a problem which causes data downloaded by the Ethernet through the HyperTransport link sometimes incorrect in the memory. As the location in memory is not random, triggers are set in the monitors to locate where the data is corrupted. Three monitors are injected between the HyperTransport PHY and controller, HyperTransport controller and Level-1 Xbar, memory controller and Level-2 Xbar just as shown in Fig.9. After compared data from the three monitors, a mismatch was found between the FPGA RAM and ASIC RAM. At last the problem is resolved by replacing the FPGA RAM with register files.

7. PERFORMANCE EVALUATION

Another goal of the MFP is to evaluate the system performance compared to the real system, but the clock frequency of memory controller is too fast compared to the one of processor core, causing the delay of memory access from the processor too short compared to a real system. To emulate the memory access delay as accurate as possible, a set of FIFO is introduced into the memory access path.

As a memory access delay in the DDR clock domain is about 30-35 clock cycles, about 70 cycles should be injected into the clock domain of process core to emulate the real memory access delay from the process core in the Multi-FPGA platform.

The impact of different memory access delay can be viewed in Fig. 10, 11 and 12. Four kind of delay are injected into the memory accessing path, including 0, 60 cycles, 70 cycles and 80 cycles.

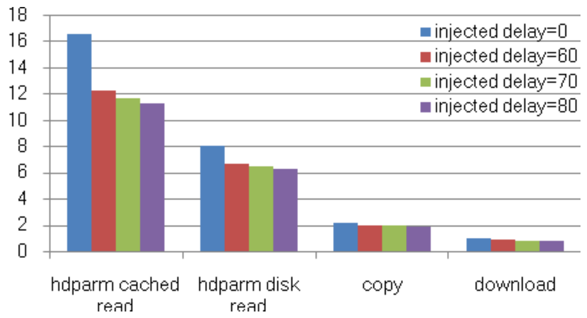

Figure 10. Bandwidth in different test (unit: MB/s)

The results of testing hdparm, copy and network download are shown in Fig.10. The results of train scale SpecCPU 2000 are shown in Fig.11 and Fig.12. It is obvious that the few cycles injected, the higher the performance is achieved in most of the cases.

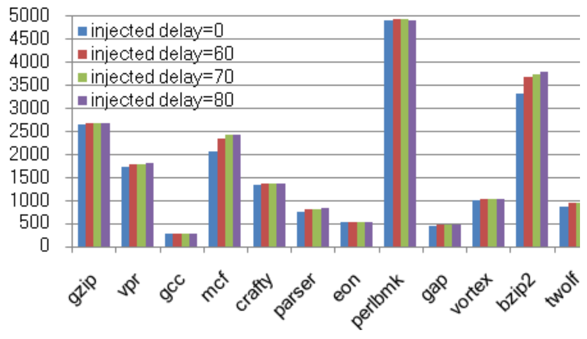

Figure 11. Execution time of Spec2000 INT (unit: Sec)

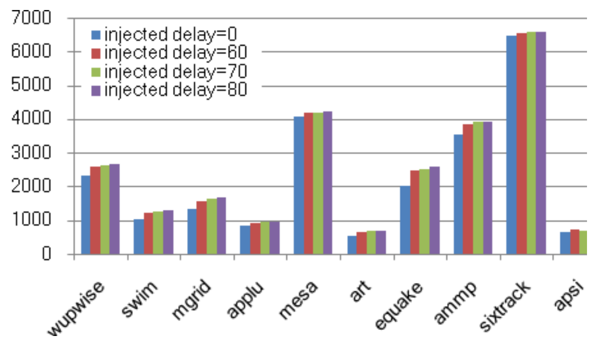

Figure 12. Execution time of Spec2000 FP (unit: Sec)

For some benchmarks in SpecCPU 2000, performance is slightly influenced by the increased delay of memory access, such as gzip, gcc, crafty, eon. That may be caused by the low miss rate in L2 cache, which alleviate the impact of increasing memory access delay.

8. CONCLUSION

As mentioned before, software simulation is time costing and impracticable for verifying and evaluating a state of art micro-processor such as Godson-2G. With shortcoming that the peripheral interfaces are hard to be verified in the hardware-based emulation, Multi-FPGA platform is developed for verification and evaluation of Godson-2G.

In this work we provide:

1) Although the current FPGA technology cannot accommodate a big design into one single device, the multi-FPGA based solution is a viable approach to emulating such a tightly-couple design as modern processors.
2) Even high-speed peripherals like HyperTransport and can be prototyped using carefully tuned FPGA.
3) Compared to post-silicon debugging, locating bugs and performance bottleneck is relatively easy by instruct a monitoring segments into the raw design.

The multi-FPGA based emulation system serves as a powerful environment for both architectural exploration and functional verification. Its capability to quickly model architectural modifications and accurately predict their performance effect to facilitate systems to chips innovations.

9. ACKNOWLEDGEMENT

Reference

[1] HyperTransport Technology Consortium. HyperTransport I/O Link Specification Revision 1.03. 2001.
[2] HyperTransport Technology Consortium. HyperTransport I/O Link Specification Revision 3.00a. 2006.
[3] Chetana N. Keltcher, Kevin J. McGrath, Ardsher Ahmed and Pat Conway. “The AMD Opteron Processor For Multprocessor Servers”. IEEE Micro, No.2, 2003.
[4] Intel Corporation. “Intel Xeon Processor 7400 Series Datasheet”. October 2008.
[5] Cadence Design Systems. “Xtreme server”. 2005. DOI=http://www.cadence.co.in/products/functionalver/xtremeserver/index.aspx
[6] Helena Krupnova and Gabriele Saucier. FPGA-based Emulation: Industrial and Custom Prototyping Solutions. In Proceedings of International Conference of Field-Programmable Logic Applications. August 2000, pp. 68-77.
[7] Cadence Design Systems. “Palladium accelerator/emulator”. 2003. DOI=http://www.cadence.com/products/functionalver/palladium/index.aspx.
[8] Ateshian, Peter and Zulaica, Daniel. “ARM Synthesizable Design with Actel FPGAs: with Mixed-Signal SoC Applications (set 3)”. 2010.
[9] Nathaniel Pinckney, Thomas Barr, Michael Dayringer, Matthew McKnett, Nan Jiang, Carl Nygaard, David Money Harris, Joel Stanley and Braden Phillips. A MIPS R2000 Implementation, DAC 2008 pp. 102-107.
[10]Zhangxi Tan, Krste Asanovic and David Patterson. An FPGA Host-Multithreaded Functional Model for SPARC v8. In Proceedings of the 3rd Workshop on Architectural Research Prototyping. 2008.
[11]Roland E. Wunderlich and James C. Hoe. In-system FPGA Prototyping of An Itanium Microarchitecture. ICCD 2004: In Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors. 2004, pp. 288-294.
[12]Rod Blaine Foist, Cristian Sorin Grecu, André Ivanov and Robin F. B. Turner. “An FPGA Design Project: Creating a PowerPC Subsystem Plus User Logic”. IEEE Transactions on Education, Volume 51, Issue 3, Aug. 2008, pp. 312–318.
[13]Shih-Lien L. Lu, Peter Yiannacouras, Rolf Kassa, Michael Konow and Taeweon Suh. An FPGA-based Pentium® in a complete desktop system. In Proceedings of the 2007 ACM/ SIGDA 15th international symposium on Field programmable gate arrays. 2007, pp. 53-59.
[14]Perry H. Wang, Jamison D. Collins, Chris T. Weaver, Belliappa Kuttanna, Shahram Salamian, Gautham N. Chinya, Ethan Schuchman, Oliver Schilling, Thorsten Doil, Sebastian Steibl and Hong Wang. Intel® atom™ Processor Core Made FPGA-Synthesizable. FPGA '09: In Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays. 2009, pp. 209-218.
[15]John Wawrzynek, David Patterson, Mark Oskin, Shih-Lien Lu, Christoforos Kozyrakis, James C. Hoe, Derek Chiou and Krste Asanovic. “RAMP: Research Accelerator for Multiple Processors”. IEEE Micro, Vol.27, No.2, 2007.
[16]Weiwu Hu, Fuxin Zhang and Zusong Li. “Microarchitecture of the Goodson-2 Processor”. Journal of Computer Science and Technology. 2005, 20(2):243-249.
[17]Weiwu Hu, Jian Wang, Xiang Gao, Yunji Chen, Qi Liu and Guojie Li. “Godson-3: A Scalable Multicore RISC Processor with x86 Emulation”. IEEE Micro, Vol.29, No.2, 2009.
[18]Weiwu Hu, Jiye Zhao, Shiqiang Zhong, Xu Yang, E. Guidetti and Christ Wu. “Implementing a 1GHz Four-issue Out-of-Order Execution Microprocessor in a Standard Cell ASIC Methodology”. Journal of Computer Science and Technology. 2007,22(1),1-14.
[19]MIPS technology. MIPS64® Architecture For Programmers Volume I: Introduction to the MIPS64® Architecture. 2008.
[20]ARM®. AMBA® AXI Protocol v1.0 Specification. 2004.
[21]Huandong Wang, Dan Tang, Xiang Gao and Yunji Chen. An Enhanced HyperTransport Controller with Cache Coherence Support for Multiple-CMP. NAS 2009. In Proceeding of IEEE

International Conference on Networking, Architecture, and Storage. 2009, pp. 215-218.
[22]Weiwu Hu, Weisong Shi and Zhimin Tang. JIAJIA: An SVM System Based on a New Cache Coherence Protocol. In Proceeding of HPCN’99, 1999, pp. 463-472.
[23]Xilinx. “Virtex-5 Multi-Platform FPGAs”. DOI=http://www.xilinx.com/products/virtex5/
[24]Synplicity Inc. “Certify”. DOI=http://www.synplicity.com/products/certify/
[25]Adrian Carbine and Derek Feltham. “Pentium Pro processor design for test and debug”. IEEE Design & Test of Computers. Vol.15, No.3, 1998, pp. 77–82.

A Multi-FPGA Based Platform for emulating a 100M-transistor-scale Processor with High-speed Peripherals

Weiwu Hu[1], Huandong Wang[1,2], Xiang Gao[1], Yunji Chen[1], Dan Tang[1,2]
[1]*Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*
[2]*Graduate University of Chinese Academy of Sciences, Beijing 100049, China*
{hww, wanghuandong, gaoxiang, cyj, tangdan, }@ict.ac.cn

ABSTRACT

Software simulation has become a powerful method in development of processors, but to simulate processors with a state of art architecture has become extremely difficult. The simulation of this kind of the processors is time-costing as the scale becomes larger and larger complied with the Moore's low. Accompanied with the enlargement of scale, many high-speed IOs are integrated into processors as SOC chips, causing another problem that the verification of high speed IOs is very limited for software simulation. With the help of FPGA based platform, the verification and evaluation of processors can be done in a relative high speed. The FPGA based platform also provides a real environment with a lot of real chips working together for the verification of high speed IOs. In this work, a multi-FPGA based platform used for verification and evaluation of Godson-2G processor is introduced. Tactics for a semi-custom partitioning and the design flow are discussed. Method to emulate DDR and HyperTransport PHYs is offered after that. Instrumentation based debugging and the performance evaluations are both presented at the end of this paper.

Categories and Subject Descriptors

C.1.0 [Processor Architectures]: General

General Terms

Design, Verification, Performance

Keywords

Godson-2G, Multi-FPGA, emulate, verification, evaluation

1. INTRODUCTION

Today's general-purpose processors have been evolving towards having more functional peripherals on-chip to provide higher performance and better cost-effective solutions. AMD has been using on-chip memory controller and HyperTransport [1] [2] links for years in its Athlon64 and Opteron [3] processors. Intel's latest Nehalem [4] processor family also departs from the traditional front side bus and integrates the memory controller into the CPU. Furthermore, other system components such as PCI-E, GPU, finding their positions in general-purpose processors has been a matter of "when", rather than "if".

With the unprecedented level of chip integration leading to much more design and verification efforts, processor designers are facing great challenges. More peripheral components added into the chip expand the already huge design space. Interactions between components get much more complicated. System-level evaluation and decision-making are suffocated by the slow simulation speeds. Meanwhile, the lack of detailed and ample peripheral modeling in the RTL-level simulation environments brings high risk into the pre-silicon verification.

To address challenges from design exploration and pre-silicon verification, more and more processor designer resort to FPGA-based prototype. As widely-deployed platforms in the area of SoCs and ASICs, the flexibility, speed, and enormous capacity qualifies them for the emulation of large and complex systems. Nevertheless, a typical state of art processor always exceeds the capacity of the largest modern FPGA device. The largest FPGAs before 2009 in production have an estimated equivalent capacity of about 1.5 million gates or 30 million transistors. Prototyping a modern processor larger than this means spreading the functionality across multiple FPGAs. It is a complicated task, but one that is well worth the effort.

This paper presents a multi-FPGA based platform (called MFP) which prototype the Godson-2G (also known as Loongson-2G), a modern superscalar processor with transistor number exceeding 100M .To the best of our knowledge, the MFP is the first platform capable of emulating a state of art processors across multi-FPGA. Taking into account the trend toward ever-increasing levels of processor integration, we view Multi-FPGA as inevitable in the long term for the pre-silicon functional verification and architectural evaluation. It's also a very promising and straightforward step to emulate huge ASICs/SOCs chips.

In the MFP system, different kinds of I/O peripheral are working together. The emulated processor core can run at up to 25 MHz. As well, the integrated DDR2/3 controller can work at 40 MHz and HyperTransport controller at 25 MHz. At such speeds, it is sufficient to boot unmodified operating system, enabling processor designer to carry out a diverse variety of architectural explorations. Time-consuming benchmarks, including both computing-intensive and IO-intensive applications, can be completely and fast evaluated. For example, the entire SPEC CPU 2K benchmark in train scale can be completed within twelve hours, in contrast to about a week by the Xtreme emulation [5], resulting in a significant improvement into the pre-silicon performance prediction.

The key contributions of the work include:

- We analyze a 100M-transistor-scale superscalar processor implementation on two virtex-5 FPGAs. Methodology to partition, synthesis, and map an almost-unmodified RTL design to a multi-FGPA target are detailed.
- We present the experiences under FPGA environment to emulate the high-speed I/O peripherals, including Hyper-Transport and DDR2/3.
- System-level Debugging, we share our methods and experience in tackling the problem at system-level developing.
- Preliminary evaluation of pre-silicon performance, including computing-intensive and I/O-intensive bench-marks validation is presented.

The remaining sections of this paper are organized as following. Section 2 summarizes related work and introduces the architecture of Godson-2G micro-processor. Section 3 describes the multiple FPGA platform used in this work. Section 4 discusses the semi-custom partitioning and design flow. Section 5 presents the method to emulate the high speed IO. Section 6 details a debugging method in FPGA and Section 7 gives some evaluation results.

2. BACKGROUND

As the scale of chip design increasing rapidly, it is difficult to evaluate a design only by software simulation. Although the emergency of multi-FPGA based [6] or hardware based emulation accelerator greatly speeds up simulation or emulation, IO behaviors and different IO specifications are still hard to verify. While the capability of a single FPGA platform is too small for emulating a 100M-transistor-scale processor, a multi-FPGA based platform is one of the best way to verify and evaluate the design of Godson-2G micro-processor.

2.1. Related Work

The three main options for evaluation and verification open to processor designers are simulation, emulation, and FPGA-based prototypes. The ever-increasing chip scale makes the software simulation notoriously slow to simulate an entire chip at RTL-accurate level. The fastest true RTL-accurate simulators of modern processors run at about 1 KHz to 10 KHz, which means two minutes of simulated time in approximately corresponding to one to ten years of simulation time. At such speeds, it is impractical to use realistic program to explore, evaluate and refine micro-architectures.

Hardware-based emulation is another reasonably popular technique, which can be viewed as a kind of accelerated and board-pluggable simulation. Several companies such as Cadence/(Xtreme,Palladium [7]), Mentor, Synopsys, EVE and Tharas sell FPGA-based accelerators, emulators or tools that take arbitrary RTL and map it to hardware to improve simulation performance. The emulators tend to be very expensive due to their complexity. With an equivalent speed of only 500 KHz to around 2MHz, however, it's still two to three orders slower than the real chip. As peripheral I/O devices in the emulation system are working at their real speed, the extreme speed imbalance between I/O peripheral and the emulated processor will introduce many odd obstacles to system-level evaluations. Taking the TCP/IP connection as an example, low frequency of the processor lead to unexpected packet losses and connection termination. Besides, I/O Bus adhere to the emulated processor should also work at a much lower speed than the sane frequency, causing I/O devices on the bus behave abnormally.

For almost all mainstream architectures, there have been FPGA-synthesizable processor designs, including ARM [8], MIPS [9], SPARC [10], Itanium [11], and PowerPC [12]. Lu et al. [13] previously presented an FPGA-synthesizable version of Intel’s vintage Pentium processor. Another notable work was the Intel Atom prototyping reported in [14], which for the first time disclosed Intel’s effort to implement an FPGA-synthesizable version of a modern x86 processor.

But the modern processor can accommodate only a fraction of the whole processor chip, making system-level validation still a dilemma.

Fortunately, Moore's law has not only enabled these dense multi-core chips, it has also enabled extremely dense FPGAs.

Compared with powerful modern processor, processor cores mentioned above are relatively simple and small. Even the biggest one of them, namely Atom, consumes a total of 47.2 million transistors, allowing the whole design being mapped into a single FPGA. Contrary to these approaches, we implement the whole processor chip into multiple FPGAs, which removes the chip-size limitation caused by the bounded capacity of a single FPGA device. Furthermore, on-chip peripheral interfaces are also integrated, making the system-level validation feasible and reliable.

The RAMP collaboration [15] is building the necessary infrastructure to compose systems of up to 1024 processors. It is targeted at facilitating software research in many-core era. The processor cores adopted by RAMP are simple in complexity and small in size. One to two dozen cores can be programmed into a single FPGA. However, it lacks the ability to validate a state of art commodity processor that exceeds the capacity of a single FPGA.

2.2. The Godson-2G Processor

The Godson project, which was initiated by ICT-CAS in 2001, was the first attempt to develop high performance general purpose processors in China [16]. Godson-2G, the latest member of Godson processors supposed to tape out in 2009, is the single core version of Godson-3 [17]. The chip measures 53.90 mm^2 in size and consumes 106.8 million transistors. The estimated peak frequency is up to 1GHz and the power dissipation is 5-7 Watt depending on applications.

The architecture of Godson-2G is shown in Fig.1. The chip is based on two crossbars with 128-bit width data buses. The Crossbar with 2x2 ports called Level-1 Xbar connects the processor core, L2 Cache and HyperTransport controller. The other one with 2x3 ports called Level-2 Xbar slots in between L2 Caches, DDR2/3 controllers and PCI/LPC interface.

The processor core of Godson-2G (named GS464) implements MIPS64 instruction set [18] in a four-issue, out-of-order execution way [19]. It fetches and decodes four instructions per cycle and dynamically issues them to five fully pipelined functional units (two fix-point, two floating-point and one memory access unit). Instructions are issued out of order by two 16-entry reservation stations and are committed in program order by a 64-entry reorder queue. Two 64-entry physical register files are used for register renaming of the general purpose and floating point registers separately. An 8192-entry pattern history table, a 9-bit global history register, a 16-entry branch target buffer and a 4-entry return address stack keep the branch history information for prediction. GS464 has a 64KB instruction cache and a 64KB data cache. A 24-

entry memory access queue that contains a content-addressable memory for dynamic memory disambiguation supports out-of-order memory access and non-blocking cache in GS464.

The Godson-2G supports a broad range of IO interfaces. These peripherals can be classified into two categories: the high speed IO and the low speed IO. High-speed IO controllers are critical for efficient executions of throughput based workloads, while low speed IO controllers provide flexibility to form a functionally complete system. The current version of Godson-2G encompasses one high-speed HyperTransport controller. The low speed IO interfaces in Godson-2G include PCI/LPC controller, UART controller, SPI controller and GPIO controller.

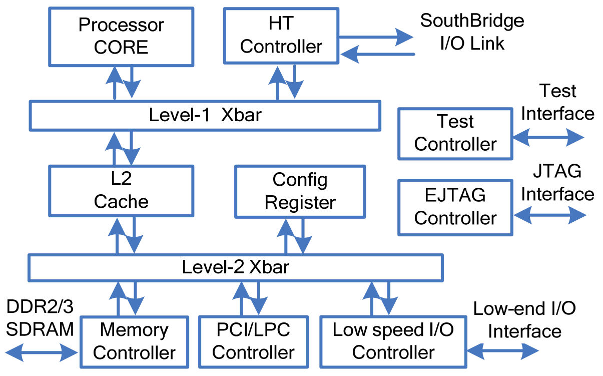

Figure 1. Architecture of Godson-2G

The two crossbars inside the Godson-2G processor are running different protocol. A cache coherence supported AXI protocol [20] is running on the Level-1 Xbar, while a standard AXI protocol is running on the Level-2 Xbar. Devices with requirement of cache coherence are placed onto the level-1 crossbar, such as L2 cache and Hyper-Transport controller. As the HyperTransport controller is placed on the Level-1 Xbar, cache coherence protocol between the native cache and IO device or remote cache in the other processors connected by the HyperTransport bus can be simply maintained. [21]

The low speed devices without cache coherence requirement are placed onto the level-2 crossbar, such as PCI/LPC controller, DDR2/3 SDRAM controller, UART controller, SPI controller, and GPIO controller. Without cache coherence protocol supported, software method can be used to maintain the coherence with caches. [22]

3. STRUCTURE OF THE PLATFORM

As a high performance SOC chip, pre-silicon validation in the FPGA platform of Godson-2G raise compelling requirements, including:

- Completeness. The capacity of the platform should be big enough to accommodate the whole chip design.
- Flexibility. The platform should be flexible for evaluating different kinds of application, including benchmarks for computing power and IO performance, running different kinds of peripheral devices.
- Applicability. Fast and accurate performance prediction for both IO-intensive and computing-intensive application.

To address these requirements and challenges, we propose a two-board solution to spread system functionality across two boards. The resulting MFP platform is constructed by a mother board and an FPGA daughter board. The mother board contains all necessary building peripheral chips and device slots, while the FPGA daughter board hosts the whole Godson-2G. To ease hardware platform setup, S2C's Dual Virtex-5 TAI Logic Module board is used as the FPGA daughter board. Two boards are erected together through high-speed PCB-to-PCB connectors. The photo of the Godson-2G MFP platform is shown in Fig. 2. The FPGA daughter board is placed above the mother board in the left.

3.1. Mother Board

The mother board contains the LPC interface part, the PCI interface part, SPI flash, UART ports and the south bridge part. LPC interface is connected to a LPC flash which stores the CPU booting codes, a nixie tube as debugging display and a super IO chip. The super IO chip is used for the function of serial port, PS/2 ports for keyboard and mouse. The PCI interface is led to two standard 32-bit PCI sockets. The LPC, SPI, UART and PCI interfaces on the mother board are directly connected to the LPC/PCI controller and the Low-speed controller in the FPGAs.

To evaluate the compatibility of the HyperTransport interface, two different mother boards are designed. The first one is using an Nvidia chipset as south bridge as shown in Fig.2, the other one is using AMD chipset, not shown here.

The chipsets in the mother boards are connected to the upper FPGA through HyperTransport interface. Most of the right half of two mother boards is occupied by various peripherals of the south bridge, including PATA socket, SATA sockets, PCI sockets, PCI-E sockets, VGA port, LPC interface and GMAC interface with a GMAC PHY chip on board.

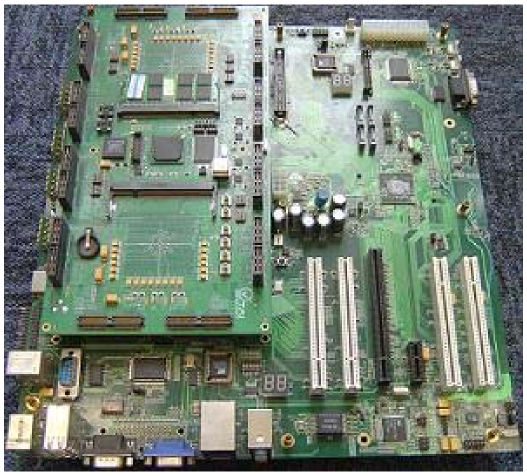

Figure 2. FPGA platform of Godson-2G

3.2. FPGA Daughter Board

The Virtex-5 330 [23] is a Xilinx FPGA with about 330K logic cells inside. The capacity of this FPGA is the largest before 2009, but still too small for this design. S2C Dual Virtex-5 330 TAI Logic Module is designed for rapid SoC/ASIC prototyping. One TAI Logic Module is equipped with two Virtex-5 330 FPGA (as shown in Fig.3, called Upper FPGA and Lower FPGA), and has two DDR2 SO-DIMM sockets for memory extensive applications. There are 600 wires connecting between the two FPGAs, and many other pins routed to twelve IO connectors distributed on the daughter boards.

Three of the IO connectors are used in this work. The one from Lower FPGA is used for LPC interfaces; two from the upper FPGA are used, one for HyperTransport link, the other one for LPC, PCI, SPI and UART interfaces.

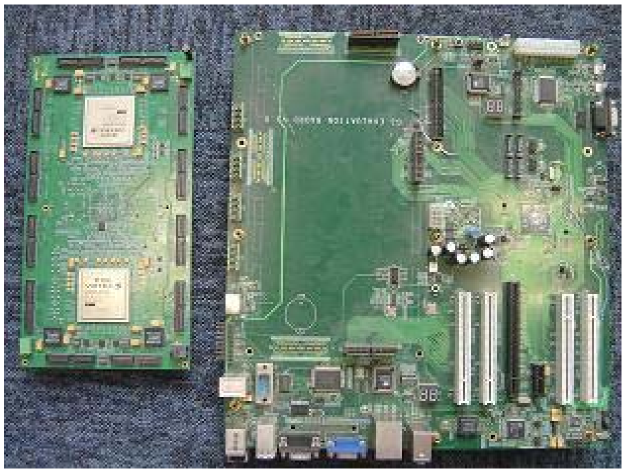

Figure 3. Boards of FPGA platform

4. PARTITIONING AND SYNTHESIZING

In the whole design flow, partitioning is the most important and complicated phase; afterward both parts of the partitioned design are synthesized separately and mapped into two FPGA chips on the same daughter board. Although EDA vendors has provided software tools such as Certify [24] to build multi-FPGA based prototypes of ASIC design in an easy, intuitive fashion, special care must be taken to avoid an inefficient solution when applying to a state of art processor like Godson-2G .

An unevenly partitioned design can easily consume 100% of the I/O resources on a device while at the same time utilizing only a relatively small amount of its internal logic resources. In the MFP implementation, we develop a semi-custom design flow. The partitioning in MFP is manually done to take into consideration the interaction between components in different FPGAs. Another purpose of manual partition is to avoid multiplex groups of IOs together, which may result in a difficult, time-consuming and cumbersome development cycle.

As for synthesizing, the RTL code of Godson-2G is entirely written in strictly synthesizable verilog language.

4.1. Partitioning Metrics

With two FPGA chips involved, it is important to partition the whole design into two parts, synthesize separately and map into corresponding FPGA. The way to partition is mainly determined by three facts. The first is resource utilization of each chip, the second is number of wires left for communication between the two chips, and the third is the location of IO interface from the mother board.

Partition may be a complicated work for most multi-FPGA designs. The worst case is that times of iteration might take place when a partition can not fit well. To avoid this problem, we do a semi-custom partition in the view of architecture. We make some modification to the original architecture of Godson-2G and partition it into two subsystems. The first one is computing subsystem with only processor core and LPC controller embedded, with the purpose to be verified independently in a single FPGA system by fetching instructions and printing debugging messages through the LPC interface. The other one is an IO and memory subsystem including the rest parts of Godson-2G.

The architecture of computing subsystem just looks like the Lower FPGA in Fig.4. Some tailored application with usage of only LPC bus such as serial port, LPC flash and nixie tube can be run on this system. The width of bus used for interconnection between the two subsystems is no more than 600, making direct communication between two FPGA possible. Direct communication proves great convenience for a multi-FPGA design, as no frequency doubling is needed, causing no frequency degression.

With full acknowledgement that the Lower FPGA with only the processor core and PCI/LPC controller embedded works well, the problem left is just to deal with the upper FPGA, which contains the IO and memory subsystem.

Considering all reasons mentioned above, architecture of Godson-2G FPGA is divided into two parts, as shown in Fig.4. Most of the IO interfaces are mapped into the Upper FPGA, with a replicate LPC controller integrated in the Lower FPGA marked as dotted lines in Fig.4, which can be bypassed when the two FPGAs working together.

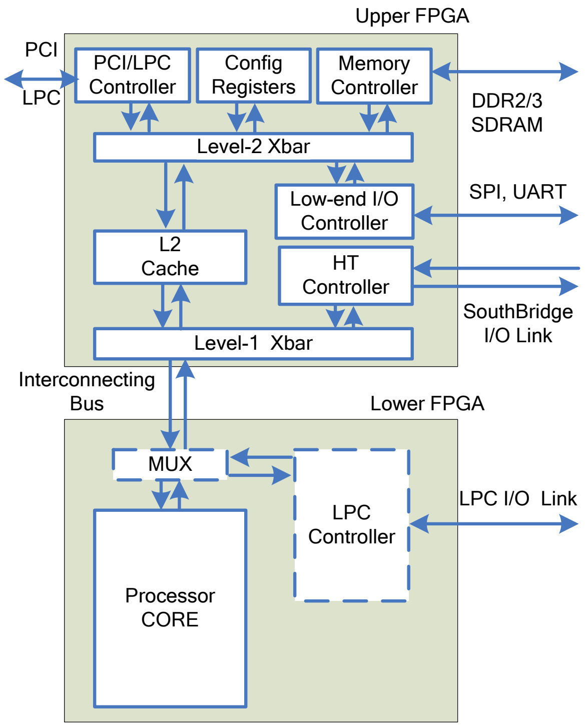

Figure 4. Architecture of Godson-2G FPGA

The interconnecting bus between two FPGA is the original processor core interface which contains no more than 600 wires. As a LPC or PCI access need not to be cached, the replicate LPC controller can be placed into the Lower FPGA and connected directly to the processor core as a slave device by a multiplexer which is enabled when debugging and disabled at normal mode. Placing the LPC controller into the Lower FPGA can make full utilization of the IO pins in Lower FPGA and also make the Lower FPGA can be verified separately and before the whole system ready but also brings a little difference from the original architecture of Godson-2G.

Thus the Lower FPGA contains a basic system which can boot up the processor core with BIOS codes stored in LPC flash and interactive through serial port by the LPC bus. The upper FPGA contains the IO subsystem, including L2 cache, memory controller and HyperTransport controller and all other interfaces of Godson-2G.

4.2. Design Flow

The entire design flow for verification and evaluation on Multi-FPGA platform is shown in Fig.5.

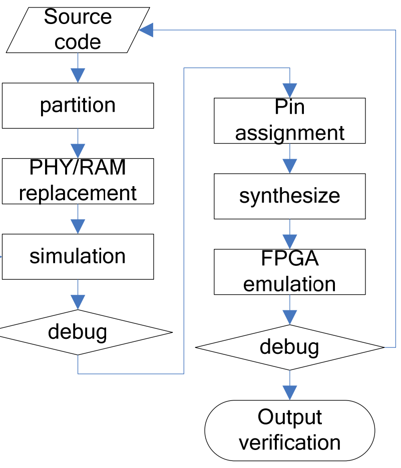

Figure 5. Flow of FPGA Verification

Original source code of Godson-2G is modified for the FPGA architecture and partitioned into Uint-1 and Unit-2 for the two pieces of FPGA. Some modules need not be verified in FPGA are removed, including the test controller, ejtag controller.

FPGA PHYs (SERDES) are used to replace the original ASIC PHYs in the design, including Hyper-Transport PHY and DDR2/3 PHY, which are hard cores in ASIC design. ASIC RAMs should be replaced by FPGA RAMs, too.

That is the all modification done to source code. Before synthesizing, source code of FPGA is first verified using simulation tools in random verification environment to avoid any low class bugs that may be injected by the modification.

All pins of the two FPGA are then allocated by their bus location and corresponding IO standard.

After all these have been done, source codes for Unit-1 and Unit-2 is then synthesized separately in ISE, a synthesizing tool for Xilinx FPGA, to generate the binary files for downloading.

A lot of applications including BIOS, Linux and different kinds of benchmarks are run and debugged on the MFP at the phase of FPGA emulation.

Resource utilizations of each FPGA are shown in Table 1.

Table 1. Resource utilization of each FPGA

	Upper FPGA	**Lower FPGA**
Logic Utilization		
Slice registers	46%	27%
Slice LUTs	65%	77%
Logic Distribution		
Occupied slices	79%	88%
IO Utilization		
Bounded IOBs	67%	43%
Block RAM	56%	78%

Utilization from both FPGA synthesizes and ASIC designs are shown in Table 2. There are some differences in the utilization between FPGA and ASIC. That many be caused by the different treatment about RAM and different cells used in FPGA and ASIC. Each slice of FPGA contains several registers and combining logic; while each instance of ASIC may be a register, a logic gate or a hard core, causing difference between the two designs.

Table 2. Resource utilization of FPGA and ASIC

	FPGA slice number	**Percent**	**ASIC instance number**	**Percent**
Processor core	93182	56%	800171	41%
HyperTransport	22041	13%	427651	22%
DDR2/3	19167	11%	405685	21%
Level-1 Xbar	13907	8%	68222	4%
Level-2 Xbar	8419	5%	56603	3%
Config registers	2102	1%	19987	1%
Level-2 cache	5883	4%	160124	8%

5. EMULATING HIGH-SPEED IO

Godson-2G is a high performance SOC with high-speed peripherals, including DDR2/3 SDRAM interface and HyperTransport interface.

One of the targets of Multi-FPGA platform is to verify the peripheral interfaces and their interactivities with the third-party chips. That's why we designed two different mother boards with different chipsets. The emulation of the DDR2/3 and HyperTransport PHY is an important step to achieve this goal. PHY (or SERDES) is a full custom designed hard core embedded on chip, used for high-speed transmission. The interface frequency of HyperTransport PHY in Godson-2G is 800MHz and DDR2/3 PHY is 400MHz. As for FPGA prototyping, this part must be re-implemented especially to emulate the function of the hard core in physical design.

The FPGA PHY of HyperTransport targets at 200 MHz and the DDR2/3 PHY targets at 40 MHz.

The main function of PHY is to convert between a set of parallel signals and a serial DDR (Double Data Rate) signal for on-board transmission. Driver of PHY is used to transform the internal parallel signals to external serial signal. Receiver of PHY is used to transform the external serial signal to internal parallel signals.

Another function is clock phase adjustment. Both output clock of driver and input clock of receiver might need to be adjusted according to the protocol.

5.1. HyperTransport Interface

The link frequency of HyperTransport PHY must be at least 200MHz due to HyperTransport specification. So that the corresponding control logic should run at 100MHz to communicate with south bridge.

As For the HyperTransport interface, ratio of parallel data to its corresponding serial signal is 4:1. For example, the waveform of driver signals is shown in Fig.6. The four parallel data from the control logic named PHY_Dn (n = 0...3) are driven by the clock named PHY_clk which is running at 100MHz. They are serialized to generate the link signal HT_cad. Clock frequency is doubled and clock phase is postponed according to the protocol requirement. For the receiver part of PHY, it just reverses the course to de-serialize the four PHY_Dn data from the external HT_cad and restores the half-frequency clock from the HT_clk.

With the implementation of FPGA PHY, frequency of HyperTransport interface is 200 MHz after system cold reset, with a link width of 8 bits. A set of asynchronous FIFO is placed between PHY and controller to transmit the data between 100 MHz clock domain of PHY and 25 MHz clock domain of the controller.

Placement of PHY can be viewed in the architecture of HyperTransport interface in Fig.7. The Tx control logic and the Rx control logic are running at 100 MHz, while the HT controller is running at 25 MHz with three sets of asynchronous FIFOs between them.

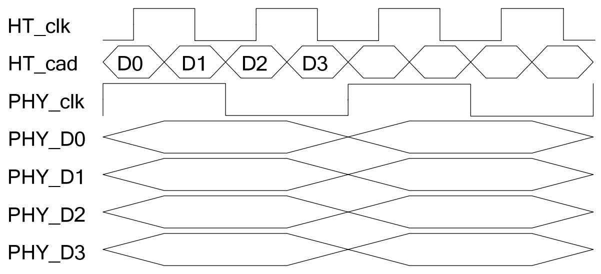

Figure 6. Waveform of HyperTransport PHY

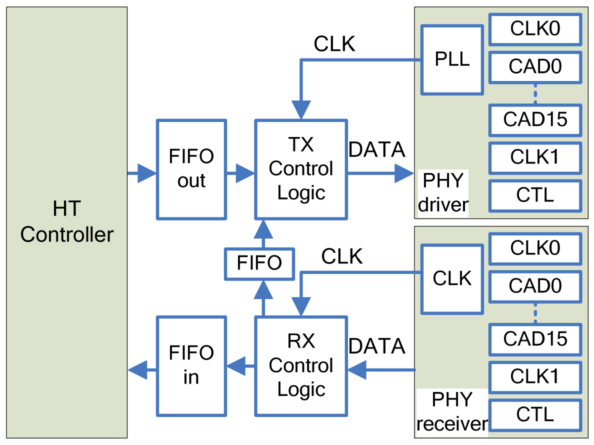

Figure 7. Architecture of HyperTransport interface

5.2. DDR2/3 Interface

Frequency requirement of DDR link is not as restrict as HyperTransport link. Thus the DDR2/3 PHY can also run at 40 MHz as DDR2/3 controller in the same clock domain.

The signal processing method of the DDR2/3 PHY is similar to HyperTransport PHY. Both them are double data rate compared to clock frequency.

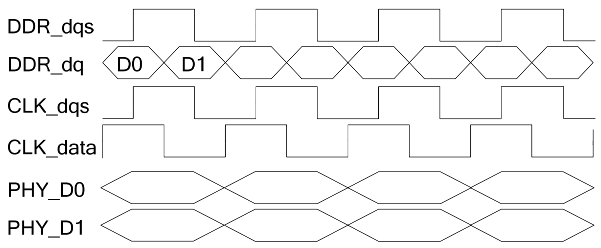

Figure 8. Waveform of DDR2/3 PHY

In order to centralize the DDR_dqs signal to the DDR_dq signals in a memory write access, two clocks are introduced into DDR2/3 PHY, as shown in Fig.8, marked as CLK_dqs and CLK_data. The latency from CLK_dqs which is responding for DDR_dqs to CLK_data which is responding for DDR_dq is one-forth clock phase.

For read dqs and read data, the DDR2/3 PHY is also responding for centralizing the positive edge of DDR_dqs to the DDR_dq signals, de-serializing and synchronizing the data to memory controller clock domain.

As there is no phase separation between read dqs and read data, read dqs must be delayed one-forth clock phase to sample and de-serialize the read data. Thus a set of delay line logic is placed into the DDR2/3 PHY for each read dqs.

6. INTEGRATION AND DEBUGGING

In MPF validation, the processor is tested in a system setting. Validation in this setting concentrates not only on the processor, but also interaction with chipset, memory system, and other peripherals.

6.1. System Environment

The configuration of processor core and peripherals are shown in Table 3. The caches are chopped to 512KB from 1MB of ASIC design to fit the capacity of FPGA.

Table 3. FPGA Configuration of Processor

	Characters	Frequency
Processor core	4 issues Out of order 64KB I-cache 64KB D-cache	25MHz
L2 cache	512 KB	25MHz
Memory controller	64 bit DDR2 1GB	40MHz
HyperTransport controller		25MHz
HyperTransport Bus	8 bit width / 16 bit width	200MHz
PCI/LPC controller		33MHz
L1/L2 Xbar	128 bit width	25MHz
Peripheral devices	10M/100M NIC on PCI GMAC on South bridge PATA on South bridge Flash, PS/2 on LPC Serial port on LPC UART port, SPI flash	

6.2. Software Stack

PMON will be first booted up on the Multi-FPGA platform after system reset. It will first initialize the processor core, caches, and then the memory controller. After that, it will initialize the HyperTransport interface and the devices on south bridge.

After everything is initialized, a network card on the PCI bus of south bridge is activated. Linux kernel is downloaded by the Ethernet link from a tftp sever.

When Linux is running, many other devices on south bridge are set up to work, including the IDE hard disk, GMAC Ethernet controller, and etc.

Many things can be done on Linux operating system. A lot of benchmarks are tested, including SPEC CPU 2000, LMbench, ftp downloading, stream test, and etc.

Some of the testing results are given in the following section.

6.3. Debugging

The best place to debug a processor bug is in the simulation environment, where all signals are available for scrutiny. Unfortunately, in the system environment of real system almost no internal signals are visible to the system debugger. A suite of tools was developed for use in the Pentium processor [25], using both architectural and micro-architectural features of the processor.

FPGA platform is also hard to debug when any abnormality emerges, as signals inside are difficult to probe. But bugs are not occasional at the very beginning. Thus the way to debug must be under consideration.

Some observing points are injected into the DMA path from HyperTransport controller to the memory controller for checking where a data corruption has happened.

Each observing point is a monitor that can be triggered by a software programmable pattern. All of the monitors are chained together in a daisy-chain topology by two sets of buses called configure bus and configure return, which are controlled by the configuration registers module as Fig.9.

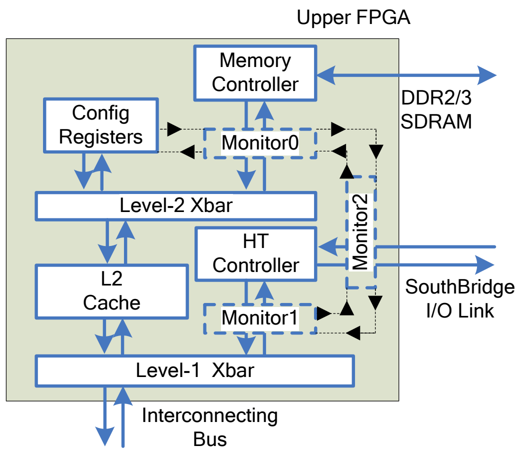

Figure 9. Injection of Monitors

The configure bus transmits the read or write command with address and write data, while configure return contains the response from the selected monitor. Software can access the monitors by accessing the configuration registers module.

With the help of injected monitors, it is easy to tell where the problem may be.

We once encountered a problem which causes data downloaded by the Ethernet through the HyperTransport link sometimes incorrect in the memory. As the location in memory is not random, triggers are set in the monitors to locate where the data is corrupted. Three monitors are injected between the HyperTransport PHY and controller, HyperTransport controller and Level-1 Xbar, memory controller and Level-2 Xbar just as shown in Fig.9. After compared data from the three monitors, a mismatch was found between the FPGA RAM and ASIC RAM. At last the problem is resolved by replacing the FPGA RAM with register files.

7. PERFORMANCE EVALUATION

Another goal of the MFP is to evaluate the system performance compared to the real system, but the clock frequency of memory controller is too fast compared to the one of processor core, causing the delay of memory access from the processor too short compared to a real system. To emulate the memory access delay as accurate as possible, a set of FIFO is introduced into the memory access path.

As a memory access delay in the DDR clock domain is about 30-35 clock cycles, about 70 cycles should be injected into the clock domain of process core to emulate the real memory access delay from the process core in the Multi-FPGA platform.

The impact of different memory access delay can be viewed in Fig. 10, 11 and 12. Four kind of delay are injected into the memory accessing path, including 0, 60 cycles, 70 cycles and 80 cycles.

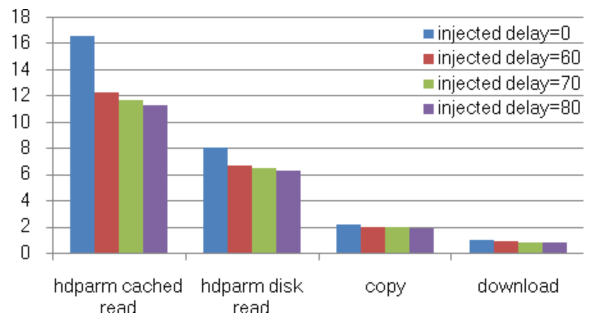

Figure 10. Bandwidth in different test (unit: MB/s)

The results of testing hdparm, copy and network download are shown in Fig.10. The results of train scale SpecCPU 2000 are shown in Fig.11 and Fig.12. It is obvious that the few cycles injected, the higher the performance is achieved in most of the cases.

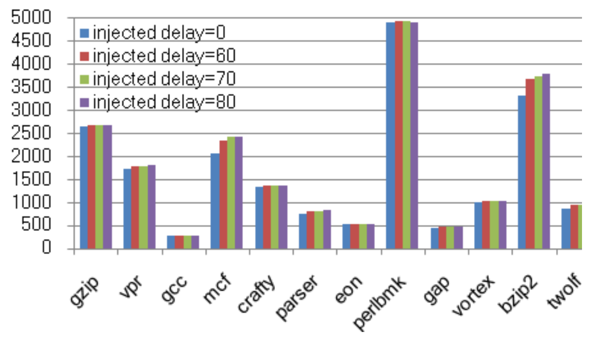

Figure 11. Execution time of Spec2000 INT (unit: Sec)

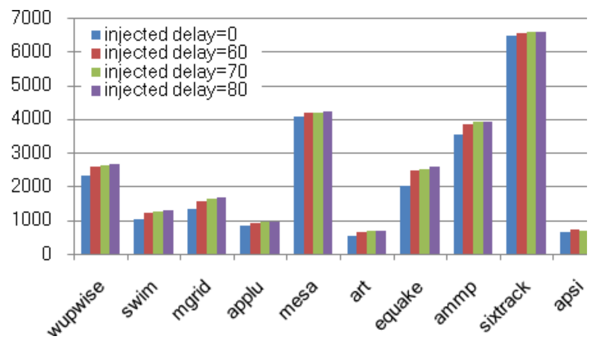

Figure 12. Execution time of Spec2000 FP (unit: Sec)

For some benchmarks in SpecCPU 2000, performance is slightly influenced by the increased delay of memory access, such as gzip, gcc, crafty, eon. That may be caused by the low miss rate in L2 cache, which alleviate the impact of increasing memory access delay.

8. CONCLUSION

As mentioned before, software simulation is time costing and impracticable for verifying and evaluating a state of art micro-processor such as Godson-2G. With shortcoming that the peripheral interfaces are hard to be verified in the hardware-based emulation, Multi-FPGA platform is developed for verification and evaluation of Godson-2G.

In this work we provide:

1) Although the current FPGA technology cannot accommodate a big design into one single device, the multi-FPGA based solution is a viable approach to emulating such a tightly-couple design as modern processors.
2) Even high-speed peripherals like HyperTransport and can be prototyped using carefully tuned FPGA.
3) Compared to post-silicon debugging, locating bugs and performance bottleneck is relatively easy by instruct a monitoring segments into the raw design.

The multi-FPGA based emulation system serves as a powerful environment for both architectural exploration and functional verification. Its capability to quickly model architectural modifications and accurately predict their performance effect to facilitate systems to chips innovations.

9. ACKNOWLEDGEMENT

Reference

[1] HyperTransport Technology Consortium. HyperTransport I/O Link Specification Revision 1.03. 2001.
[2] HyperTransport Technology Consortium. HyperTransport I/O Link Specification Revision 3.00a. 2006.
[3] Chetana N. Keltcher, Kevin J. McGrath, Ardsher Ahmed and Pat Conway. “The AMD Opteron Processor For Multprocessor Servers”. IEEE Micro, No.2, 2003.
[4] Intel Corporation. “Intel Xeon Processor 7400 Series Datasheet”. October 2008.
[5] Cadence Design Systems. “Xtreme server”. 2005. DOI=http://www.cadence.co.in/products/functionalver/xtremeserver/index.aspx
[6] Helena Krupnova and Gabriele Saucier. FPGA-based Emulation: Industrial and Custom Prototyping Solutions. In Proceedings of International Conference of Field-Programmable Logic Applications. August 2000, pp. 68-77.
[7] Cadence Design Systems. “Palladium accelerator/emulator”. 2003. DOI=http://www.cadence.com/products/functionalver/palladium/index.aspx.
[8] Ateshian, Peter and Zulaica, Daniel. “ARM Synthesizable Design with Actel FPGAs: with Mixed-Signal SoC Applications (set 3)”. 2010.
[9] Nathaniel Pinckney, Thomas Barr, Michael Dayringer, Matthew McKnett, Nan Jiang, Carl Nygaard, David Money Harris, Joel Stanley and Braden Phillips. A MIPS R2000 Implementation, DAC 2008 pp. 102-107.
[10]Zhangxi Tan, Krste Asanovic and David Patterson. An FPGA Host-Multithreaded Functional Model for SPARC v8. In Proceedings of the 3rd Workshop on Architectural Research Prototyping. 2008.
[11]Roland E. Wunderlich and James C. Hoe. In-system FPGA Prototyping of An Itanium Microarchitecture. ICCD 2004: In Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors. 2004, pp. 288-294.
[12]Rod Blaine Foist, Cristian Sorin Grecu, André Ivanov and Robin F. B. Turner. “An FPGA Design Project: Creating a PowerPC Subsystem Plus User Logic”. IEEE Transactions on Education, Volume 51, Issue 3, Aug. 2008, pp. 312–318.
[13]Shih-Lien L. Lu, Peter Yiannacouras, Rolf Kassa, Michael Konow and Taeweon Suh. An FPGA-based Pentium® in a complete desktop system. In Proceedings of the 2007 ACM/ SIGDA 15th international symposium on Field programmable gate arrays. 2007, pp. 53-59.
[14]Perry H. Wang, Jamison D. Collins, Chris T. Weaver, Belliappa Kuttanna, Shahram Salamian, Gautham N. Chinya, Ethan Schuchman, Oliver Schilling, Thorsten Doil, Sebastian Steibl and Hong Wang. Intel® atom™ Processor Core Made FPGA-Synthesizable. FPGA '09: In Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays. 2009, pp. 209-218.
[15]John Wawrzynek, David Patterson, Mark Oskin, Shih-Lien Lu, Christoforos Kozyrakis, James C. Hoe, Derek Chiou and Krste Asanovic. “RAMP: Research Accelerator for Multiple Processors”. IEEE Micro, Vol.27, No.2, 2007.
[16]Weiwu Hu, Fuxin Zhang and Zusong Li. “Microarchitecture of the Goodson-2 Processor”. Journal of Computer Science and Technology. 2005, 20(2):243-249.
[17]Weiwu Hu, Jian Wang, Xiang Gao, Yunji Chen, Qi Liu and Guojie Li. “Godson-3: A Scalable Multicore RISC Processor with x86 Emulation”. IEEE Micro, Vol.29, No.2, 2009.
[18]Weiwu Hu, Jiye Zhao, Shiqiang Zhong, Xu Yang, E. Guidetti and Christ Wu. “Implementing a 1GHz Four-issue Out-of-Order Execution Microprocessor in a Standard Cell ASIC Methodology”. Journal of Computer Science and Technology. 2007,22(1),1-14.
[19]MIPS technology. MIPS64® Architecture For Programmers Volume I: Introduction to the MIPS64® Architecture. 2008.
[20]ARM®. AMBA® AXI Protocol v1.0 Specification. 2004.
[21]Huandong Wang, Dan Tang, Xiang Gao and Yunji Chen. An Enhanced HyperTransport Controller with Cache Coherence Support for Multiple-CMP. NAS 2009. In Proceeding of IEEE

International Conference on Networking, Architecture, and Storage. 2009, pp. 215-218.
[22]Weiwu Hu, Weisong Shi and Zhimin Tang. JIAJIA: An SVM System Based on a New Cache Coherence Protocol. In Proceeding of HPCN'99, 1999, pp. 463-472.
[23]Xilinx. "Virtex-5 Multi-Platform FPGAs". DOI=http://www.xilinx.com/products/virtex5/
[24]Synplicity Inc. "Certify". DOI=http://www.synplicity.com/products/certify/
[25]Adrian Carbine and Derek Feltham. "Pentium Pro processor design for test and debug". IEEE Design & Test of Computers. Vol.15, No.3, 1998, pp. 77–82.