Parallel Processing Using FPGAs

Shaswot Shresthamali

Department of Electronics and Communication Engineering,
Sagarmatha Engineering College, Tribhuvan University
Sanepa, Lalitpur

shaswot@sagarmatha.edu.np

***Abstract*—This document introduces FPGAs (Field Programmable Gate Arrays) and highlights their parallel processing capability. FPGAs are reprogrammable hardware chips for implementation of digital logic. Their reconfigurability is attributable to the presence of configurable logic slices and interconnects. The possibility of hardware configuration in FPGAs allows the designer to design multiple modules that run parallel and independently to each other. This paper discusses how FPGAs offer true hardware parallelism and deliberates on some areas which benefit from FPGAs' parallel processing such as DSP (Digital Signal Processing), Data Acquisition and Processing, Text Parsing and Image/Video Processing. To demonstrate the possibility and the consequent advantages of parallel processing, a matrix multiplier was designed for a Spartan-3E FPGA with the help of High Level Synthesis (HLS) tools. Two possible solutions, with and without parallel processing, were obtained which are briefly discussed here.**

***Index Terms*—FPGA, Parallel Processing, Spartan-3E FPGA, DSP, HLS.**

I. Introduction

The sheer volume of information to be processed in today's world has risen exponentially with time. This has led to the development of faster processor cores with higher clock speeds. Now the paradigm is shifting to multi-core processors. A good example of this would be Intel's i3, i5 and i9 processors with multiple cores on a single chip.

Although parallel processing is possible using multiple cores, their cost and power consumption are major issues in their practical application. Achieving parallel processing using a single microprocessor with a single core is an avenue that has been extensively researched. Numerous pipelining algorithms have been developed to achieve the fastest throughput – however, the fact remains that a single processor can execute only one instruction at a time.

In contrast, FPGAs (Field Programmable Gate Arrays) can offer true parallelism in processing. They can be configured to have multiple processors functioning in parallel.

Another fundamental difference between FPGAs and microprocessor processing is in the approach to processing. Microprocessors are almost exclusively based on the Von Nuemann architecture or the Harvard architecture. Both these architectures have a common feature in that the processors have to fetch data *and* the instructions to process the data i.e. they use a stored program concept. However, FPGA processing is purely hardware based processing – there is no fetching of instructions required; the program description determines the configuration of the hardware itself i.e. the processing is implemented in hardware rather than software.

A. Introduction to FPGA

FPGAs are configurable silicon chips that can be configured to mimic any type of digital circuitry. They consist of CLBs (Configurable Logic Blocks) and highly flexible programmable interconnects.

The CLBs consist of LUTs (Look-Up Tables), wide multiplexers and storage elements (usually D flip-flops). The LUTs (usually with three to five inputs) can be configured to perform any Boolean function. The use of multiplexers allow for combination of LUTs to increase the input width of the CLB.

The interconnections between different CLBs is accomplished using an interconnect matrix or a global routing matrix. These interconnect matrices are configurable and allows for high flexibility in the interconnection between different CLBs.

FPGAs also come with configurable IOBs (Input Output Blocks) to comply with a large number of standards like LVTTL, LVCMOS, HSTL etc.

Modern FPGAs have other silicon hardcores built in to facilitate better designs and higher speeds. For instance, the Spartan-6 FPGAs from Xilinx consist of DCM (Digital Clock Manager), block RAM, gigabit transceivers, and the DSP48A1 slice in addition to the CLBs and IOBs. It supports upto 147K logic cell density, 3.2 Gigabits/sec integrated serial transceivers.

FPGAs are configured using HDL (Hardware Description Language) such as Verilog, VHDL or System Verilog. The HDL is synthesized and then implemented onto the FPGA using mapping and PAR (Place and Route) tools.

Another method for generation of HDL codes is via HLS. The generation of synthesizable HDL codes from high level languages such as C, C++ or System C is termed as HLS. Xilinx Vivado HLS 2013.4 is one such HLS tool. Xilinx Vivado HLS 2013.4 is used to convert high level descriptions of a design into Register Transfer Language (RTL) netlist which is then implemented on the FPGA.

The use of HLS for generation of synthesizable HDL allows the designer to create multiple 'solutions' for a defined design objective, allowing the designer to explore design trade-offs and arrive at an optimal solution.

II. Parallel Processing Using FPGAs

FPGAs exhibit true parallel process execution. Each independent task is assigned to a different area of the chip. Each task performs autonomously without having to share resources with other simultaneous processes. Adding more parallel threads will not affect any of the existing processes either. The amount of parallelism in FPGA is limited only by the amount of physical chip space available.

A. Hardware Flexibility

Implementing designs in FPGAs means that the designer is designing the hardware using FPGA. They are not restricted to any predetermined hardware architecture or function. An FPGA allows the designer to program product features and functions, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed in the field – hence the name "field-programmable".[1]

Since FPGA configuration implies actual hardware configuration, the designer has many options on which to customize the design. The I/O pins can be configured according to the designer's convenience. Even the data bus width can be made arbitrarily wide depending upon the design. The FPGA Architecture provides the flexibility to create a massive array of application specific ALUs that enable both instruction and data level parallelism. Because data flows between operators, there are no inefficiencies like processor cache misses; FPGA data can be streamed between operators. The parallelism offered by FPGA architecture can be easily seen in HPC (High Performance Computing) – relevant parameters like

- Internal ALU bandwidth is in the order of terabytes/sec (TB/sec)
- Integer operation throughputs are in the order of Tera-operations/sec (TOPS)
- Floating point operations are in the order of gigaflops/sec (GFLOPS) [2].

B. FPGAs as Co-processors

Instead of completely replacing CPUs, FPGAs can also play the role of co-processors. The use of such co-processors can unload some of the burden from CPUs for specific processing tasks. FPGAs are attractive co-processors because of the potential for tailored design and parallelism. FPGAs are also very interesting in regard to power consumption as they consume significantly less power, yet have performance comparable to conventional CPUs. This makes FPGAs good candidates for implementing cores in multi-core systems where certain data processing tasks can be off loaded from the main CPU [3].

C. Design Trade-offs

FPGAs have uncommitted logic resources along with others that can help developers directly steer module-to-module hardware infrastructure and trade off resources and performance by selecting the appropriate level of parallelism to implement an algorithm [4]. One can trade between speed and power consumption when designing with FPGAs. This makes it very popular in low power applications.

D. System Latency

Use of FPGAs also decreases a lot of latency time that is inevitable in processors. Significant amount of time is consumed when moving data between memories (like RAM and cache) and between ALU and memory. The cycles used in processors for data transfer between memory elements and ALU can be utilized effectively for processing in FPGAs.

FPGAs can bypass a lot of other system latency. With a conventional processor, a system might be receiving data via a TCP socket onto an Ethernet chip, but that then has to go through a MAC layer, then a North Bridge chip, then onto the processor main bus, then an interrupt has be flagged, then all the data has to be transferred into the user space. All these things can of course be done very quickly but there are nonetheless a lot of steps to go through that do not apply to FPGAs [5].

III. Application of FPGA Parallel Processing

FPGAs were originally used as glue logic but now their functionality and application has taken giant strides in the recent years. Their parallel processing capability has made them a popular tool in various areas.

A. Digital Signal Processing(DSP)

DSP requires massive amounts of parallel computation. This makes it very suitable for its implementation in FPGAs. Now-a-days, most FPGAs have built in DSP slices to assist in DSP. Some of the major areas where DSP can be implemented using parallel processing of FPGAs are discussed as follows.

1) MAC (Multiply and Accumulate) Blocks: DSP use MAC blocks extensively. A DSP expression usually is of the form:

$$Y[k] = \sum c[k].x[k]$$

The multiplication of a coefficient $c[n]$ with an input sequence $x[n]$ is accomplished by using a shift-and-add approach in microprocessors. An FPGA outperforms a DSP processor by performing multiplication and accumulation steps in parallel. Even more optimized implementation of MAC can be achieved by using distributed arithmetic [6] – [8] and FPGAs.

2) FFT: FFT that is accomplished using butterfly techniques benefit from using parallel processing. The various butterfly stages can be computed in parallel to get to the result faster.

High-speed FFT cores have become an essential requirement for real time spectral monitoring and analysis. As the monitoring bandwidth grows with higher frequency spectrums, systems must be designed to convert time domain signal to frequency domain faster – necessitating faster FFT operations. In most modern systems, parallel FFTs run parallel at extremely high sample rates (12.5 Gigasamples/sec) taking advantage of wideband A/D converters.

3) Digital Filters: Digital filtering requires significant amounts of multiply/accumulate operations. FIR and IIR filters use shift registers and tap delays that can be easily realized using the resources in FPGAs. Direct Form I and Direct Form II realization of IIR filters benefit immensely from parallel processing capability of FPGAs.

B. Data Acquisition/Logging

(DASs) Data Acquisition Systems consist of multiple inputs that need to be processed. A processor is unable to capture the data from various inputs and sensors simultaneously. It uses pipelining and memory buffering to achieve this. In addition, each different input will require a different type of processing – hence the processor will have to load new instructions for each sensor's input data.

Using FPGAs can overcome many of the shortcomings of using a single processor for DAS systems. The FPGA can be configured to have independent signal processing ALUs for each of the sensor inputs. In addition, the large number of pins available in an FPGA combined with its hardware configurability allows it to accept and process the signals from multiple sources simultaneously.

C. Text Parsing

The parallelism of FPGAs is perfectly suited for high speed text parsing where multiple newsfeed must be screened for a huge number of keywords.

Using processors to achieve this would be quite inefficient and would require multiple processors. This would not only increase the cost of the system but also the power consumption.

As the number of keywords to be identified increases, the processor performance degrades appreciably as compared to FPGAs. While precise figures vary with implementation, a CPU will take approximately 100 times longer than FPGAs to parse a text for fifty expressions. [5]

D. Image/Video Processing

Although the power of CPUs has increased, image and video processing still require algorithms that are accelerated by DSP, RISC and FPGAs.

Image and video data occupies a large bandwidth and usually requires parallel processing. Dataflow architectures are usually based on this requirement and are designed so as to unburden the image and video processing load from the CPU. Processes such as filtering, colour correction and noise suppression are computationally intensive, requiring high bandwidth and parallel processing. This can be implemented using FPGAs as coprocessors to CPUs to deliver faster results and more efficient use of resources.

IV. RESULTS

To demonstrate the advantages of using FPGAs for parallel processing, a 32-bit signed integer 3x3 matrix multiplication was implemented in a Xilinx Spartan-3E XC3S5000E-FG320-4 FPGA.

Xilinx Vivado HLS 2013.4 was used to generate the RTL description followed by its synthesis. Two different solutions were generated for the possible synthesis and implementation of the design. In solution 1, no pipelining was implemented. The second solution, solution 2, used the pipeline directive to ensure maximum hardware parallelism during the implementation of the hardware. The synthesis estimates are illustrated in Tables I, II and III.

It is very clear from the synthesis estimates that the latency of the multiplier has been reduced dramatically from 133 clock cycles in the first solution to just nine clock cycles in the second solution (Table II). This has been achieved at the cost of more hardware. For instance, while only three multipliers are employed in the first solution, the second solution employs 81 multipliers (Table III).

In short, in Solution 2, the use of multiple multipliers and other associated hardware in parallel allowed for faster computation of results.

TABLE I
TIMING ESTIMATES

Clock		Solution 1	Solution 2
Default	Target	20.00 ns	20.00 ns
	Estimated	16.94 ns	16.94 ns

TABLE II
LATENCY ESTIMATES

		Solution 1	Solution 2
Latency	Minimum	133 clock cycles	9 clock cycles
	Maximum	133 clock cycles	9 clock cycles
Interval	Minimum	134 clock cycles	6 clock cycles
	Maximum	134 clock cycles	6 clock cycles

TABLE III
RESOURCE UTILIZATION ESTIMATES

	Solution 1	Solution 2
18K Block RAM	0	0
Flip Flops	155	1445
LUTs	137	1638
18x18 Multipliers	3	81

V. CONCLUSION

The use of FPGAs for implementing digital systems provides designers with a high degree of flexibility. A designer who implement designs in FPGAs has the advantage to reconfigure the hardware in order to trade between performance and resource utilization depending upon the design resources and constraints. Power consumption, processing speed and chip area can each be traded for another to come to an optimized solution.

Computer architecture is changing towards heterogeneous multi-core systems to achieve computational parallelism on

the massive amounts of input data. This is a challenge for both hardware and software designers.

FPGAs have a considerable share of the silicon market and are finding more areas of applications every day. FPGAs will dominate a lot of the applications where application specific ALUs will be required. Their hardware configurability and parallel computational capability will play a critical role in the implementation of multi-core computational paradigm.

FPGAs already have demonstrated their success in application specific processing systems owing to their parallelism and high speed computation. Although FPGAs may not replace CPUs completely, they will play a very important role as co-processors to increase the computational ability of processors.

ACKNOWLEDGMENT

I wish to acknowledge the Department of Electronics Engineering at Sagarmatha Engineering College, Sanepa for providing me with the resources to help me prepare this paper.

I am grateful towards Mr. Deepesh Man Shakya for his advice, support and encouragement.

I also wish to thank Mr. Dil Bahadur Chettri for his valuable comments.

REFERENCES

[1] Altera Website on FPGA. [Online]. Available: http://www.altera.com/products/fpga.html

[2] Prasanna Sundararajan, *High Performance Computing Using FPGAs*.

[3] Rene Mueller, Jens Teubner and Gustavo Alonso, *Data Processing on FPGAs*.

[4] Shuai Chey, Jie Liz, Jeremy W. Sheaffery, Kevin Skadrony and John Lachz, *Accelerating Computer-Intensive Applications with GPUs and FPGAs*

[5] Automated Trader Website. [Online]. Available: http://www.automatedtrader.net/articles/spotlight/514/fpgas-_-parallel-perfection.html

[6] G.R. Goslin, *A Guide to Using FPGAs for Application-Specific Digital Signal Processing*. Available: www.xilinx.com/appnotes.dspguide.pdf

[7] L. Minster, *The Role of Distributed Arithmetic in FPGA-based signal processing*, Available: http://home.att.net/~pcuenin/theory1.pdf

[8] B. New, *A distributed arithmetic approach by designing scalable DSP chips*, EDN, Aug. 17, 1995.

Shaswot Shresthamali was born in Kathmandu, Nepal, in 1989. He received his B.E. degree in electronics and communication engineering from Tribhuvan University, Nepal in 2012. Following his graduation, he joined the Department of Electronics and Communication Engineering in Sagarmatha Engineering College as an Assistant Lecturer. In 2013, he started 'Xilinx Research Laboratory' in Sagarmatha Engineering College of which he is also the Lab Coordinator. He is presently an Assistant Professor and the Research Coordinator of Sagarmatha Engineering College.

His current research interests include Audio Digital Signal Processing, FPGAs, Programmable Logic and VLSI Design. He is also a member of IEEE Solid State Circuits Society as well as the IEEE Robotics and Automation Society.