Implementation of an 8x8 Discrete Cosine Transform on Programmable System-on-chip

Bryan Chan Jia Ching, Ab Al-Hadi Ab Rahman, Nabihah Ahmad

Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Skudai, Malaysia
Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Malaysia

E-mail: icebryanchan@gmail.com, hadi@fke.utm.my, nabihah@uthm.edu.my

Abstract. This paper presents experimental results that compares between a full software (SW) implementation and a software/hardware (SW/HW) co-design implementation of a DSP algorithm on a Xilinx Zynq programmable System-on-chip (SoC). The case study being used is the 8x8 two-dimensional discrete cosine transform (2D DCT), present in the popular JPEG and MPEG4-AVC encoder. The full SW design is implemented on a hardcore ARM processor on the FPGA SoC. The SW/HW co-design utilizes both the ARM processor and the Configurable Logic Blocks (CLB) of the FPGA SoC, where the communication channel is implemented using the Xillybus FIFO buffers, implemented as an external DRAM. In this case, the core 2D DCT operations are executed on the CLB, while data initialization and transfers are implemented on the processor. Results show that SW implementation is faster compared to SW/HW implementation for data inputs of less than 0.48 megapixels. As data inputs get larger, SW/HW implementation shows better performance, with up to 2x faster for 2 megapixels data input size. This study proves the viability of implementing the 2D DCT operations as dedicated hardware accelerator in multimedia encoders.

1. Introduction

The trend in systems design is gearing towards the use of heterogeneous system-on-chip (SoC) where a general purpose processor (GPP) is integrated as a single chip with custom hardware accelerators. Modern FPGAs today provide the capability to program the SoC for both the custom hardware parts and the software part running on the GPP. In such case, the FPGA is commonly referred to as a programmable SoC. Major FPGA vendors from Intel (formerly Altera) and Xilinx today also provide specialize FPGA chips that consist of hardcore ARM processors with configurable logic blocks (CLB) in a single chip.

Compared to classical full hardware implementation on FPGAs, programmable SoCs give flexibility to designers to implement non speed critical or very complex parts of the design in SW, while allowing inherently parallel parts of the code to be implemented as custom logic for increased performance. As designs today become more complex, full hardware implementation may not be feasible as time to market becomes shorter. HW implementation is known to be difficult and more challenging compared to SW implementation, but usually achieves better performance. Programmable SoCs give a nice balance for designers in terms of performance and flexibility.
This paper explores SW/HW Co-design methodology on a programmable SoC. Some part of the code runs on SW, while some others on HW. Both of the GPP and CLB resources are used in the FPGA chip. Co-design methodology overcomes the complexity faced from full HW implementation. However, the challenge in co-design is establishing the handshake between HW and SW, as well as managing the data transfer between them. One of the major issues in co-design is the communication cost between the two platforms, whereby the throughput of the system may be limited. Co-design is often suggested to be used in large data processing or those with intensive calculation requirements and may not be suitable for systems that require large and continuous data transfers [1].

This paper presents experimental results using the Zedboard from Digilent [2]. The board contains a single programmable SoC called the Zynq-7000 that consists of two ARM CPUs and the CLB for custom HW. There are several works reported in literature on the use of Zynq SoCs, but none validated on a streaming signal processing application, as proposed in this work. The work in [3] presents an implementation of a hadoop cluster for data clustering; [4] profiles the performance of the ARM cortex A9 on the Zynq; [5] and [6] present a co-design architecture for Blokus Duo Solver and SHA-3 encryption/decryption respectively.

The application proposed for the co-design implementation is the two dimensional discrete cosine transform, usually referred to as the 8x8 2D DCT. The algorithm is implemented in C code using the ISO/IEC 23002-2 algorithm [7]. Xilinx Vivado HLS is used to convert the C code to Verilog for HW implementation. The ARM CPU on the Zynq is used mainly for data initialization, transfers, and validation, while the core operation is implemented on the CLB. Data transfer buffer between SW and HW is implemented using Xillybus [8]. This co-design methodology is then compared with a full SW implementation running only on the GPP in terms of performance.

2. Background

2.1. 2D Discrete Cosine Transform

Discrete Cosine Transform (DCT) is a mathematical function that is mostly used in image and video compression, i.e. JPEG, MPEG-4 AVC/H.264, and HEVC encoders. DCT transforms data from time domain into frequency domain for removing data redundancies in a lossy compression, as down sampling and quantization is irreversible [9]. A fast DCT is necessary for encoding or decoding images with large resolutions or videos with high frame rate [10].

The 2-D DCT calculation has a high degree of computation complexity [11]. There are many algorithms to compute the 2-D DCT coefficients [12] [13] and the algorithm chosen in this paper is based on the algorithm defined by ISO/IEC 23002-2 standard [7]. This algorithm has been simplified in [14], and shown in Figure 1. In an 8x8 input matrix or input block, the first 1-D DCT is running row-wise while the second 1-D DCT is running column-wise based on the outputs from the first 1-D DCT. This simple pipelined algorithm reduces the complexity of the calculation factor by four [13].

3. SW and SW/HW Implementation Methodology

3.1. Zedboard and Xilinx Zynq-7000

Xilinx Zynq-7000 Programmable SoC consists of one embedded chip of a dual-core ARM Cortex-A9 MP Core microprocessor and a FPGA Fabric. At the board level, it can be divided into two parts, namely Processor System (PS) and Programmable Logic (PL). The dual-core ARM processor runs at a maximum frequency of 677MHz while the maximum frequency of PL or FPGA Fabric is determined by the design or by default it is 100MHz [2]. PS is made up of ARM microprocessor, SDRAM, flash memory controllers, peripherals as well as general-purpose input/output (GPIO) . Zedboard can be used to connect with a host machine through Universal
Figure 1. Fast 1D DCT algorithm defined in the ISO/IEC 23002-2 standard

Asynchronous Receiver/Transmitter (UART) or through a memory card with bitstream file generated by Vivado software [15].

3.2. **HW/SW Co-design using Xillybus**

Xillybus IP is used to connect AXI ports and FIFOs with I/O devices on linux based operating system on the Zynq called Xillinux. In a SW/HW co-design methodology, when an application
logic is called, the processor passes the configuration header and messages to the hardware accelerator to compute the message.

3.3. Xillybus IP Core
Xillybus is a Direct Memory Access (DMA)-based IP core for data transport in between a FPGA and a host machine. In other words, Xillybus is a bridge for Processor System (PS) and Programmable Logic (PL) for data transfer. Xillybus supports wide range of products and board including Altera and Xilinx devices. FPGA application logics connects to the Xillybus through standard FIFOs. The Xillybus IP core then connects directly to the PS. Data transfers are made simple since PS handles the files in and out from PL through the Xillybus. The operation is represented in Figure 2. On host machine or PS side, there is no need for software library, any programming language can be used to access Xillybus streams without any new specific extensions. As shown in the figure, the PS is interfaced to the PL via PCIe bus connected to the Xillybus IP core. The core in turns are interfaced to application FIFO, where by default are implemented as external DRAM. The FIFOs connect directly to the PL.

![Figure 2. Xillybus IP core](image)

3.4. Xillinux by Xillybus
Xillinux is a Linux distribution with additional feature like FPGA code kit. Just like any other Linux distribution, Xillinux allows Xilinx products, including but not limited to Zedboard, to run a graphical desktop with the support of keyboards, mouse, monitor screen and many other input output devices. The main feature or purpose of this Linux distribution is to make the integration between Processing System (PS) and Programmable Logic (PL) simple and easy. Xillinux uses SD card as its hard disk and it comes with a built-in PL development kit. The development kit allows Xillybus to be piped with the PS (ARM processors for Zedboard). Application logic can be connected to the FIFOs in PL and plain files can be sent in and out using simple operations from Xillinux.

3.5. Wrapper Function for the PL
Wrapper function is responsible for the interface between the PS and the PL. Inside the wrapper function, pragma directives are used to interface with the Xillybus IP core. Wrapper function works like the main function in a simple C program which will later make calls to the calculation function which is called the synthesized function. Programmer or user needs to decide how a normal program will separate into software part and hardware part. For the hardware part of a program, it will be included in the synthesized function. This part of the program will be running in the PL as the hardware part of the software-hardware co-design. In this work, the whole 2-D DCT will be put inside the synthesized function to maximize the performance. The
wrapper function together with the synthesized function will then be debugged and synthesized by Vivado HLS. After that, the synthesized hardware code will then be exported into RTL in dcp format which will later be combined with Verilog code of the provided firmware for Xillinux, in order to generate a bitstream file. The bitstream file be will copied into the SD card where the Operating System of Xillinux is installed. When the Xillinux is booted up with the new bitstream file, the hardware code is ready to be used. Any changes to be made on the hardware part of the code must be done starting from the step of designing the wrapper and synthesized function.

3.6. Host Program for the PS

After the PL is ready with the desired synthesized function, a host program is written and compiled in PS to communicate and work with the function inside the PL. The PS and PL are interconnected with two standard FIFOs, one for sending data from PS to PL and another is vice versa. Therefore, it is wise to make use of basic parallel programming which includes a simple forking in the host program. The host program is forked into two processes, which are writing and reading data. This will prevent the processing from stalling due to lack of data to process or output data to be cleared up. Arrays are also used to send blocks of data into PL for processing and receive processed blocks of data from PL. These approaches will reduce but not totally eliminate the I/O overhead as well as the impact of software-hardware latencies.

The host program written in this work reads a text file containing the input blocks and then sends to PL for DCT before outputting the resulting blocks into a new text file. The host program or the software part of the program is responsible for reading inputs and writing outputs in the software environment. A detailed flowchart is shown in Fig. 3.

![Figure 3. Host program interfacing with the PL](image-url)
3.7. In-code Performance Profiling
In order to compare the performance between the SW/HW co-design program and the full software designed program, a C program with exactly the same functions is written and compiled in the Xillinux environment.

In this work, in code performance profiling is used to ensure the accuracy of the result because an external profiler might cause a heavy load on the operating system. Therefore, to ensure an unbiased profiling, the timing interval must be logical.

Since both of the programs are having the same functionality, it is safe to put the profiling interval as the beginning of the main function and just before the main function ends. The resulting time taken for executions is then noted down for further analysis.

3.8. Results and Analysis
Both the full software program and SW/HW co-design programs are executed for different numbers of 8x8 blocks. Numbers of blocks are set to 100, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 12000, 14000, 16000, 18000, 20000, 22000, 24000, 26000, 28000, 30000, 32000 and 34000 respectively. Each execution is repeated for at least 100 times in order to obtain the average execution time for each number of blocks processed.

The results are plotted in the graph as shown in Figure 4. Note that the graph is not using a constant scale for the x-axis.

![Graph of time taken to execute the DCT for different number of 8x8 blocks](image)

**Figure 4.** Graph of time taken to execute the DCT for different number of 8x8 blocks

As shown by the graph, both programs show linear relationship between the number of blocks processed and the time taken to complete the given tasks. As the number of blocks increase, the time taken to complete the DCT of the blocks increases linearly.

Initially, when the number of blocks fed into the DCT for both programs are below 7500 blocks, the co-design program shows relatively slower performance up to 1 second difference than that of full software designed program. This is mainly due to the input/output (I/O) communication time is relatively large when handling small number of blocks. Besides, an idle function of 1 second is included in the co-designed program in order to prevent the Xillybus devices from stalling and it is necessary to keep the program functioning as expected.
However, both of the line graphs crosses over at approximately 7500 of blocks where the performance of co-design program has caught up on the performance of software program. At this number of blocks processed, the time taken for I/O has become more insignificant compared to the performance of Programmable Logic.

Both line graphs continue to increase linearly with a constant gradient respectively after exceeding 7500 blocks of data processed. However, the performance difference between both programs is becoming bigger. The gradient of full software program execution time (0.213 milliseconds/block) is greater than that of software-hardware co-designed program (0.077 milliseconds/block). This means when the amount of input or blocks in this case increases, the advantage of using software-hardware co-design is becoming more significant.

Thus, from the result shown, co-design is suitable for programs that deal with large amount of data. In this work, less than 7500 blocks of data are considered as small amount and the DCT design should be done just by using full software design.

However, nowadays with the advanced growing technology in image processing and cameras, even smartphone cameras are generally above 8 megapixels. This means JPEG encoder nowadays requires the capability to handle more than 125,000 blocks of data if a block contains 8x8 of pixels.

Figure 5 illustrates clearly how the co-design program outperforms the full software designed program. The full software designed program shows a constant throughput of 0.3 megapixels per second despite the increase of pixels to be processed. However, the throughput of co-design program increases linearly over the increase of pixels to be processed up to 0.58 megapixels/second at 2 megapixels. Figure 5 also shows that the throughput of co-design program outperforms that of full software designed program once the processed pixels exceed 480,000 pixels or 0.48 megapixels.

![Figure 5. Graph of throughput vs number of pixels.](image_url)

Resource utilization on the PL side is shown in Figure 6. As can be seen, the resource utilization is very low since the design size is relatively small. The power consumption can be expected to be low as well due to low resource usage.
4. Conclusion
In this paper, we provide experimental results and insights on SW/HW co-design versus full SW implementation of a 8x8 DCT on Xilinx Zynq programmable SoC. The DCT is a very important process in image and video compression technologies such as in JPEG and MPEG, and known to be the most compute intensive part. In order to speed up the process compared to full SW implementation, the C code of the DCT has been synthesized to Verilog and implemented on HW, while data initialization, transfers, and validation on SW. Results show that due to communication cost, SW/HW implementation outperforms full SW implementation for data size larger than 0.48 megapixels. Results also reveal performance improvement of roughly 2x for 2 megapixel input size for DCT. Future work is to implement the full JPEG encoder using the SW/HW co-design methodology and validate the viability of using the CLB for acceleration.

Acknowledgment
The authors would like to thank UTM for providing the facilities, and the Ministry of Education for providing the funds for this research (Vot No. 4F659).

Reference
[1] Bezati E, Yviquel H, Raulet M and Mattavelli M 2011 A unified hardware/software co-synthesis solution for signal processing system Proceedings of the 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP) p1
[2] Xilinx 2017 ZedBoard Technical Specifications Internet: http://zedboard.org/content/zynq-fpga-fabric-clock-speed
[3] Lin Z and Chow C 2013 Zcluster: A zynq-based hadoop cluster Field Programmable Technology (FPT) International Conference on, IEEE. p450
[4] Powell A and Silage D 2015 Statistical performance of the ARM cortex A9 accelerator coherency port in the xilinx zynq SoC for real-time applications ReConFigurable Computing and FPGAs (ReConFig) International Conference on p1
[5] Sugimoto N and Amano H 2014 Hardware/software co-design architecture for Blokus Duo solver Field Programmable Technology (FPT) International Conference on p358
[6] Ahmed K E and Farag M M 2015 Hardware/software co-design of a dynamically configurable SHA-3 System-on-Chip (SoC) Electronics, Circuits, and Systems (ICECS) p617
[7] ISO/IEC 2006 Information technology âŤ MPEG video technologies âŤ Part 2: Fixed-point 8x8 IDCT and DCT transforms ISO/IEC 23002-2
[8] Xilinx 2017 FPGA coprocessing for C/C++ programmers (part II) Internet: http://xillybus.com/tutorials/vivado-hls-c-fpga-howto-2
[9] Miano J 1999 Compressed Image File Formats âŤ JPEG, PNG, GIF, XBM, BMP Addison Wesley Longman Inc USA
[10] Chiper D F 2016 A new VLSI algorithm for a high-throughput implementation of type IV DCT *International Conference on Communications (COMM)* p17

[11] Arai Y, Agui T and Nakajima M 1988 A Fast DCT-SQ Scheme for Images *Transactions of IEICE* E71 1095

[12] Kovac M and Ranganathan K 1995 JAGUAR: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard *Proceedings of the IEEE* 83 247

[13] Bhaskaran V and Konstantinides K 1999 Image and Video Compression Standards Algorithms and Architectures åÅ Second Edition Kluwer Academic Publishers USA

[14] Zabidi N N and Ab Rahman A A 2017 VLSI Design of a Fast Pipelined 8x8 Discrete Cosine Transform *International Journal of Electrical and Computer Engineering (IJECE)* 7 1430

[15] Xilinx 2013 Xilinx Zynq-7000 All Programmable SoC Overview Xilinx Inc