Scalable FPGA Architecture for DCT Computation Using Dynamic Partial Reconfiguration

Jian Huang, Matthew Parris, Jooheung Lee, and Ronald F. DeMara
School of Electrical Engineering and Computer Science, University of Central Florida
Orlando, FL 32816 USA

Abstract - *In this paper, we propose FPGA-based scalable architecture for DCT computation using dynamic partial reconfiguration. Our architecture can achieve quality scalability using dynamic partial reconfiguration. This is important for some critical applications that need continuous hardware servicing. Our scalable architecture has two features. First, the architecture can perform DCT computations for eight different zones, i.e., from 1×1 DCT to 8×8 DCT. Second, the architecture can change the configuration of processing elements to trade off the precisions of DCT coefficients with computational complexity. Using dynamic partial reconfiguration with 2.1 MB bitstreams, 16 distinct hardware architectures can be implemented. We show the experimental results and comparisons between different configurations using both partial reconfiguration and non-partial reconfiguration process.*

Keywords: DCT, dynamic partial reconfiguration, FPGA

1. Introduction

Scalable architecture for video coding is particularly suitable for applications that deliver the contents to a wide range of terminals and use a complex and heterogeneous delivery network, such as broadband video distribution and mobile communications. Discrete cosine transform (DCT) is widely used in image/video coding standards, such as JPEG, MPEG-1/2/4, H.261/3/4 [1]. Due to the high computational complexity of DCT [2], hardware implementation has been preferred.

Compared to Application Specific Integrated Circuits (ASIC) design, Field Programmable Gate Array (FPGA) design is more flexible, and has fast development time and low Non-Recurring Engineering (NRE) cost. Dynamic partial reconfiguration is to reconfigure a part of the FPGA while the other parts of the FPGA are still operational and the chip is active [3]. So, the functions that the FPGA is performing will not be interrupted during the reconfiguration. This is very important for some applications such as medical image diagnosis and military related video processing. It also has many advantages, such as increased resource sharing and utilization.

In this paper, we present FPGA-based scalable architecture for 2D-DCT computation using dynamic partial reconfiguration. Our scalable 2D-DCT architecture is based on distributed arithmetic (DA) [4]. DA architecture is suitable for FPGA implementation due to its ROM-based computations of inner products. Our scalable architecture is working in two different modes to reduce the computational complexity and the power consumption. First, it can achieve quality scalability by performing 2D-DCT operations for different zones, i.e., from 1×1 to 8×8, as shown in Fig. 1. Only the DCT coefficients in the shaded area are computed. Our scalable architecture can be adjusted through dynamic partial reconfiguration to perform different types of DCT zonal coding. Second, our scalable architecture can be reconfigured to reduce the precision of DCT coefficients especially when quantization parameter increases to achieve high compression ratio. Third, our scalable architecture can reconfigure the unused DCT computation module for other functions, such as motion estimation computation.

There are some previous works related to scalable DCT. Kim and Yoo used eight masks for DCT to scale the complexity [5]. Xanthopoulos and Chandraka used MSB rejection and row-column classification to reduce the power consumption [6]. Kinane et al. used clock gating to implement a variable length 1D-DCT for shape adaptive coding [7].

The rest of the paper is organized as follows. In Section 2, we present our proposed scalable architecture using dynamic partial reconfiguration. In Section 3, we show the experimental results and comparisons. In Section 4, we briefly conclude our work.

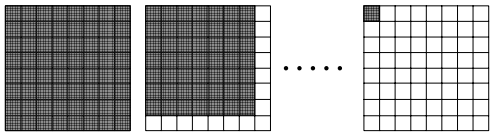

Fig. 1. Different Zones for DCT Computation

2. Architecture

The computational complexity can be reduced by decomposing the 2-D DCT into two 1-D DCT computations together with a transpose memory. We use Chen's algorithm for the 1-D DCT implementation [8], as shown in (1).

$$\begin{bmatrix} F(0) \\ F(2) \\ F(4) \\ F(6) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix} \cdot \begin{bmatrix} f(0)+f(7) \\ f(1)+f(6) \\ f(2)+f(5) \\ f(3)+f(4) \end{bmatrix}, \quad \begin{bmatrix} F(1) \\ F(3) \\ F(5) \\ F(7) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & -G & E \\ G & -F & E & -D \end{bmatrix} \cdot \begin{bmatrix} f(0)-f(7) \\ f(1)-f(6) \\ f(2)-f(5) \\ f(3)-f(4) \end{bmatrix} \quad (1)$$

$$A = \cos\frac{\pi}{4}, B = \cos\frac{\pi}{8}, C = \sin\frac{\pi}{8}, D = \cos\frac{\pi}{16}, E = \cos\frac{3\pi}{16}, F = \sin\frac{3\pi}{16}, G = \sin\frac{\pi}{16}.$$

Distributed arithmetic (DA) is a bit-serial operation that performs inner-product computations. We implement the DA architecture in a parallel fashion to improve the performance [4].

2.1 Top Level Architecture

The top level architecture for scalable DCT is shown in Fig. 2. There are eight reconfigurable areas in the design, i.e. PE0, PE1, PE2, PE3, PE4, PE5, PE6, PE7. These eight reconfigurable areas can be dynamically reconfigured using partial reconfiguration. There are two ways of reconfiguration of the PEs to achieve quality scalability of DCT computations. First, the PEs can be removed or added using dynamic partial reconfiguration to achieve zonal coding for the DCT coefficients, as illustrated in Fig. 1. The zones for the DCT coefficients vary from 8×8 to 1×1. For example, if only 4×4 DCT coefficients are needed, PE0, PE1, PE4, and PE5 are configured for the computations. Later, if 8×8 DCT coefficients are needed, PE2, PE3, PE6, and PE7 can be added using dynamic partial reconfiguration. Second, the internal logic of PEs can also be changed through dynamic partial reconfiguration to reduce the precision of resultant DCT coefficients. This is reasonable because quantization truncates some of the LSB bits of DCT coefficients. There are two main advantages of our scalable architecture using dynamic partial reconfiguration. First, the DCT computations are not interrupted when switching from different zones or different precisions. Second, the unused PEs can be utilized by reconfiguration for other functions, such as motion estimation computation.

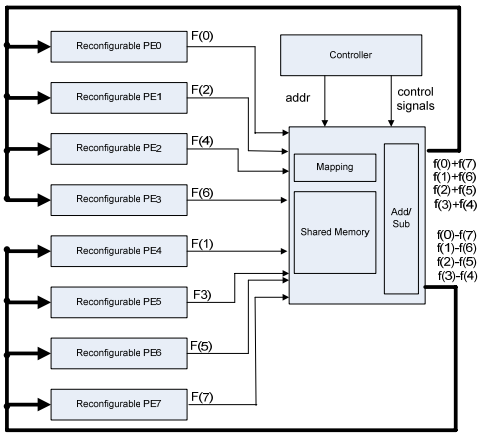

Fig. 2. Top level architecture of scalable DCT

The controller is used to generate the address and control signals for data fetching and data assigning. The shared memory, data mapping, and butterfly addition/subtraction are combined together. Mapping block is to read the data from the memory or write the data to the memory according to the address and control signals generated by the controller module. Addition/subtraction performs butterfly addition or subtraction to generate the input data for the PEs.

2.2 Reconfigurable PE

The schematic diagram of the reconfigurable PE is shown in Fig. 3. The DCT coefficients here are 12-bit width. The inputs of the PE have 13-bit width due to the butterfly additions and subtractions. We use 13 ROM Shifters (RS) in each PE to parallelize the inner product computations. Each RS has 4-bit width input, i.e., one bit plane of x1_in, x2_in, x3_in, and x4_in. For example, RS0 accepts the MSB bit plane, and RS12 accepts LSB bit plane. Therefore, 1-D DCT can be performed in one clock cycle. The result from each PE is 12-bit width DCT coefficient. The RS consists of 4 Exclusive-ORs, one 8-word ROM, one adder, one shifter, and one initial condition register. The ROM contents and initial register value are shown in Fig. 3. *A1*, *A2*, *A3*, and *A4* are four values which can be obtained from each row of the coefficient matrix in Equation (1). We also truncate the ROMs to explore the trade-off between the power consumption and the precision.

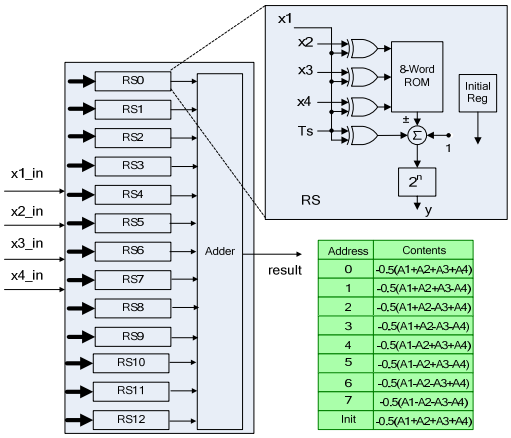

Fig. 3. Schematic diagram of PE

3. Experimental results

Using the Xilinx Early Access Partial Reconfiguration (EAPR) design flow [9], the scalable architecture is implemented on the Xilinx Virtex-4 SX35 Video Starter Kit. The scalable architecture previously described naturally allows each PE to reside within a separate reconfiguration area for modification of its configuration without disturbing the remaining portion of the FPGA. Fig. 4 shows the implementation of the scalable architecture with the locations of the eight reconfiguration areas.

Partial reconfiguration allows flexibility in selecting the quality of precision of a specific PE along with the total number of PEs allocated to the DCT application. Each reconfigurable region is able to implement one PE. In 8x8 2D-DCT computations, for example, each reconfigurable area is configured to contain one PE each, totaling 8 PEs. In 1x1 computations, one reconfigurable area contains one PE while the other 7 reconfigurable areas are made available to other video functions such as motion estimation. In our experiment, three types of PEs are designed: a full precision DCT PE, a partial precision DCT PE, and an Empty PE. The Empty PE allows those reconfiguration areas not being used by the 2D-DCT computations to contain no switching logic to reduce dynamic power consumption.

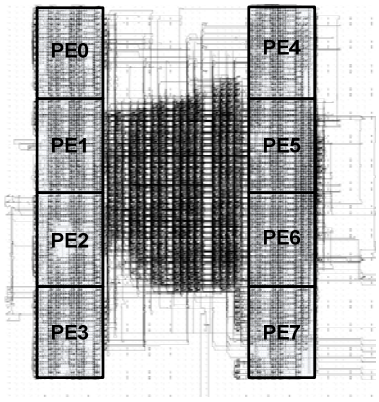

Fig. 4. Location of 8 PEs on V4SX3

Since the Full Precision PE is the largest of the three configurations, its resource requirements determine the size of the reconfiguration areas. The Virtex-4 architecture has a configuration frame resolution of 16 CLBs in height— reduced from the Virtex-2 architecture whose configuration frame resolution includes the entire height of the device [10]. Therefore, the reconfiguration areas span the minimum of 32 slices in height, whereas the width of each reconfiguration area is minimized to encompass its specific PE design.

A partial bitstream is generated for each reconfiguration area and for each type of PE. For example, 24 partial bitstreams are generated in our implementation of 8 reconfiguration areas and 3 types of PEs. The partial bitstreams generated for each of the Full Precision PEs range from 22,306 bytes to 28,306 bytes. Because the Full Precision PE represents the largest slice utilization, its bitstream sizes are the upper bounds for all types of PEs. For comparison, a bitstream file size of an Empty PE is 10,586 bytes. Before partial bitstreams are used, the FPGA is initialized first with a full bitstream. In designing the initial full bitstream, the user determines the most useful combination of type and number of PEs as the initial configuration of the FPGA— full or partial precision, and 1x1, 2x2, etc. The size of the intial bitstream is always 1,712,614 bytes, regardless of whether all 8 Full Precision PEs are implemented or only 1 Full Precision PE with 7 Empty PEs are implemented. In comparison to a full bitstream, partial bitstreams are significantly smaller, reducing the storage space required to store the various bitstreams. The results show that the file size of a Full Precision PE bitstream is about 1.6% of a full bitstream.

Table I lists a comparison between one non-partial reconfiguration scenario and two partial reconfiguration scenarios. In the case of non-partial reconfiguration, a full bitstream needs to be generated and stored for each 2D-DCT configuration. For example, a full bitstream of 1,712,614 bytes is required for a 1x1 Full Precision DCT configuration. To implement an 8x8 Full Precision DCT function, another full bitstream is required. To implement a 4x4 Full Precision DCT function with 4 Motion Estimation PEs, a third full bitstream is required. For three distinct hardware arrangements, 4.9 MB of storage space is required. To switch between each of these hardware arrangements, the entire FPGA is reconfigured, stopping all video processing elements. The shortest configuration time needed to switch between hardware arrangements is also the worst at 17 ms. The configuration time is estimated based on the timing of SelectMAP using continuous data loading [11], as shown in (2).

$$T_{config} = (bytes + 3) \cdot \frac{1}{f_{cclk}} \qquad (2)$$

Here, bytes is the number of bytes of the bitstream stored in the external PROM and f_{cclk} is the clock frequency of the SelectMap configuration clock set to 100 MHz in our estimations.

TABLE I
SIZES AND CONFIGURATION TIMES OF BITSTREAMS

		Bitstream (bytes)	Configuration Time (ms)
Non-PR	1x1 Full 2D-DCT	1,712,614	17 ms
	4x4 DCT & 4 ME PEs	1,712,614	17 ms
	8x8 Full 2D-DCT	1,712,614	17 ms
	3 H/W Arrangements (Best/Worst Config. Time)	4.9 MB	17ms/17ms
PR	Initial (8x8)	1,712,614	17 ms
	8 Full Precision PEs	226,448	0.28 ms each
	8 Partial Precision PEs	226,448	0.28 ms each
	8 Empty PEs	84,688	0.11 ms each
	16 H/W Arrangements (Best/Worst Config. Time)	2.1 MB	0.11/2.24 ms
PR	Initial (8x8)	1,712,614	17 ms
	8 Full Precision PEs	226,448	0.28 ms each
	8 Partial Precision PEs	226,448	0.28 ms each
	8 Empty PEs	84,688	0.11 ms each
	8 Motion Estimation PEs	226,448	0.28 ms each
	80 H/W Arrangements (Best/Worst Config. Time)	2.3 MB	0.11/2.24 ms

In an implementation of the scalable architecture using partial reconfiguration, a user stores at least one initial configuration bitstream and all partial bitstreams on an external ROM. In calculating the storage requirements, the worst-case Full Precision PE partial bitstream filesize— 28,306 bytes— is used for partial bitstream totals. The total space required for implementing the initial bitstream and all three types of 2D-DCT PEs—Full, Partial, and Empty— is approximately 2.1 MB. For this small amount of required storage, 16 distinct hardware arrangements are possible. Switching between these hardware arrangements does not disturb logic residing outside of the reconfiguration areas. The shortest configuration time to switch between arrangements is 0.11 ms by implementing one Empty PE, for example, to switch from 8x8 DCT to 7x7 DCT. The longest configuration time is estimated to be 2.24 ms to switch, for example, from 8x8 Partial Precision to 8x8 Full Precision.

In our future work, we plan to further expand the scalable architecture by adding motion estimation configuration to our

current architecture. The addition of eight motion estimation PE bitstreams would only increase the storage requirement by 0.2MB while increasing the number of possible hardware arrangements from 16 to 80.

We simulate full precision and partial precision PE designs using QCIF frame format (176×144). We compare the precisions between double precision floating point calculations and hardware calculations using mean square error (MSE). The precision comparisons are shown in Fig. 5. QP is the quantization parameter used in MPEG-2 system [12]. While the QP increases, the MSE decreases.

The power estimation comparisons and the throughput comparisons are shown in Fig. 6. Based on our experiments, it takes 2N+23 clock cycles to perform the DCT computations for an N×N zone, and N×N cycles to output the DCT coefficients. We can see from Fig. 5 and Fig. 6 that our architecture provides the scalability among precision, power, and throughput.

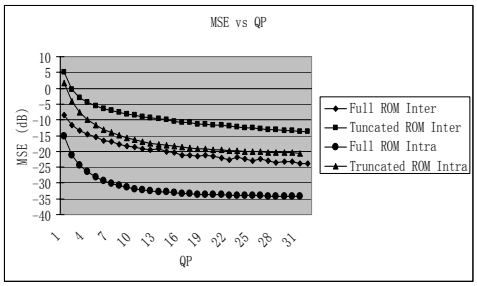

Fig. 5. Precision comparisons

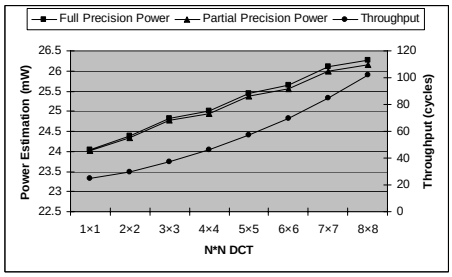

Fig. 6. Power and throughput comparisons

4. Conclusion

In this paper, we have presented our exploration of scalable architecture of DCT computations using FPGA dynamic partial reconfiguration. We used distributed arithmetic based architecture for DCT computations. Using dynamic partial reconfiguration, the processing elements of the DCT architecture can be changed on the fly including the number of the PEs and the internal logic of the PEs. The FPGA does not need to be stopped while changing the configuration, which is important for many image/video applications. We provided detailed implementation results and comparisons for different configurations of PEs using both partial reconfiguration process and non-partial reconfiguration process.

5. References

[1] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding*, Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] I.E.G. Richardson and Y. Zhao, "Adaptive algorithms for variable complexity video coding," *Proc. IEEE ICIP*, vol. 1, pp. 457-460, Oct., 2001.

[3] Cindy Kao, "Benefits of Partial Reconfiguration," *Xcell Journal*, fourth quarter 2005.

[4] Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *IEEE ASSP Magazine*, July 1989.

[5] H.C. Kim and K.Y. Yoo, "Complexity-scalable DCT-based video coding algorithm for computation-limited terminals," *IEICE Trans. Commun.*, vol. E88-B, No. 7, July 2005.

[6] T. Xanthopoulos and A. P. Chandrakasan, "A Low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *IEEE Journal of Solid-state Circuits*, vol. 35, no. 5, May 2000.

[7] A. Kinane, V. Muresan, N. O'Connor, N. Murphy, and S. Marlow, "Energy-efficient hardware architecture for variable N-point 1D DCT," *in proc. PATMOS 2004 - IEEE International Workshop on Power And Timing Modeling, Optimization and Simulation*, pp. 780-788, 2004.

[8] W. H. Chen, C. Smith, and S. Fralick, "A fast computation algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, pp. 1004-1009, 1977.

[9] Early Access Partial Reconfiguration User Guide, Xilinx Inc., San Jose, CA, 2006.

[10] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfiguration of Xilinx FPGAs," *in Proc. Field Programmable Logic and Applications*, 2006.

[11] XAPP138 - Virtex FPGA series configuration and readback, Xilinx Inc., San Jose, CA, 2005. [online]. Available: http://www.xilinx.com/support/documentations/aplication_notes/xapp138.pdf

[12] Haskell Barry, *An Introduction to MPEG-2*, Chapman & Hall, 115 Fifth Avenue, New York, NY, 1996.