DIGITAL SIGNAL PROCESSING AND DIGITAL SYSTEM DESIGN USING DISCRETE COSINE TRANSFORM

Chia-Jeng Tseng and Maurice F. Aburdene

Department of Electrical Engineering
Bucknell University
Lewisburg, PA 17837

ABSTRACT

The discrete cosine transform (DCT) is an important functional block for image processing applications. The implementation of a DCT has been viewed as a specialized research task. We apply a micro-architecture based methodology to the hardware implementation of an efficient DCT algorithm in a digital design course. Several circuit optimization and design space exploration techniques at the register-transfer and logic levels are introduced in class for generating the final design. The students not only learn how the algorithm can be implemented, but also receive insights about how other signal processing algorithms can be translated into a hardware implementation. Since signal processing has very broad applications, the study and implementation of an extensively used signal processing algorithm in a digital design course significantly enhances the learning experience in both digital signal processing and digital design areas for the students.

1. INTRODUCTION

In electrical and computer engineering, digital signal processing and digital design are two important and closely related areas. Digital signal processing is devoted to the theory and applications of Fourier transforms. Common applications include image and speech processing. Digital design studies the issues in Boolean algebra, logic optimization, and digital implementation. Instruction in these two subjects is generally offered independently. We believe that the integration of signal processing techniques and digital design methodologies profoundly enhances students' learning effectiveness and instructors' teaching experience [4].

The DCT has been widely adopted in digital signal processing applications such as image compression and speech processing. The hardware design of the DCT is considered to be a specialized research topic [2, 3, 6]. We believe that the hardware implementation of a DCT algorithm is a valuable component for learning digital design.

A micro-architectural model, which defines a digital circuit as a data part and a control part, is a powerful concept for implementing a digital design. This model has been extensively used to automate the design of digital circuits in the field of high-level synthesis [7].

The methodology was used in an "Advanced Digital Design" course for electrical engineering seniors and graduate students. The lectures focused on VHDL, logic optimization, and high-level synthesis. The three-hour weekly laboratory emphasized the design and implementation of the DCT and two other projects using a field-programmable gate-array (FPGA) board [8].

In the following sections, we describe how we apply general digital design methodologies to the implementation of an efficient DCT. Various methods for teaching students about system partitioning, design integration, design space exploration are also discussed in detail.

The paper is organized as follows. Section 2 describes a recursive DCT algorithm. Section 3 discusses the methods of generating a micro-architectural implementation of the algorithm. The issues of various design considerations for further optimization and design space exploration are addressed in Section 4. Section 5 presents the techniques for testing and debugging a design. Finally, Section 6 summarizes the lessons and conclusions from this study.

2. A DCT ALGORITHM

A DCT maps a set of sampling data in the time or space domain into a corresponding set of data in the frequency domain. Extensive research shows that high compression ratio can be obtained for image data in the frequency domain. Let x and Y represent the sampling data in the space or time domain and the transformed data in the frequency domain, respectively. Also, let N be the number of data points. Equation (1) and Equation (2) define the recursive DCT algorithm described in [1]. Equation (1) is used to determine the a_{N-1} and a_{N-2} for each k; the range of k is from 0 to $N-1$. Equation (2) is for calculating $Y(k)$.

$$a_{-2} = a_{-1} = 0$$

$$a_i = 2\cos(\theta_k)\, a_{i-1} - a_{i-2} + x(i) \quad (1)$$
$$for\ i\ =\ 0, 1, ..., N\text{-}1.$$

$$Y(k) = \frac{2}{N}\gamma_k (-1)^k \cos\left(\frac{\theta_k}{2}\right)[a_{N-1} - a_{N-2}] \quad (2)$$

The remaining parameters in Equations (1) and (2) are defined as:

$$\gamma_0 = \frac{1}{2}$$
$$\gamma_k = 1 \qquad for\ k = 1, 2, ..., N\text{-}1.$$

$$\theta_k = \frac{k\pi}{N} \quad for\ k\ =\ 0, 1, ..., N\text{-}1.$$

3. BUILDING A MICRO ARCHITECTURAL DESIGN FOR DCT

Equation (1) and Equation (2) are expressed as a procedure shown in Figure 1 for direct hardware implementation. The expressions in the bold phase show the control flow. The computation of the first two terms is instantiated. At the beginning, a_0 and a_1 are stored in a_{i-2} and a_{i-1}, respectively. The remaining part is kept in the inner loop. The outer loop specifies the computation of the *Y* data set.

```
for k = 0 to N - 1 do
{
    a0 = x(0);
    a1 = x(1) + 2x(0) cos(π / N);
    for i = 2 to N - 1 do
    {
        ai = [2 cos(θk)]ai-1 - ai-2 + x(i);
    }
    Y(k) = [ (2/N) γk (-1)^k cos(kπ / 2N) ] (aN-1 - aN-2);
}
```

Figure 1: DCT Algorithm

Having transformed the original algorithm into a form suitable for hardware implementation, we make the following high-level design decisions.

- Scheduling
  One consideration for maximizing parallelism is to unroll the *for* loops. Since the computation of each iteration in the inner loop depends on the results of the two previous iterations, we decided to leave the loop structure intact. Also, we assume that *N* is equal to eight for processing 8x8 image blocks.
- Data path design
  Based on the above procedural description, the modules needed in the data paths include multipliers, *a*-coefficients generators, and *Y*-generators. Registers, arithmetic and logic units, and interconnections are required to build the data paths. The registers are labeled as a_{i-2}, a_{i-1} and a_i. Additional registers labeled *temp1* and *temp2* are introduced to serve as buffers for computation. The data operators required for computing the *a*-coefficients include a multiplier, a subtractor, and an adder. Each data transfer infers an interconnection variable. Various design styles for the synthesis of interconnection units may be considered. For example, in the multiplexer style, the interconnections with the same destination are combined into a multiplexer. In the bus style, the interconnections sharing common sources and/or destinations are merged to generate a bus structure. The interconnections shown in Figure 2 are either in the form of a wire or a multiplexer. The clique-partitioning procedure described in [7] is applied to the synthesis of these data path components. The data paths of the *Y*-generator, shown in Figure 3, are synthesized by the same method.
- Controller design
  In addition to the normal sequential state transitions, there are two nested loops in the algorithmic description. The inner and outer loops calculate the *a*-coefficients and the *Y*-outputs, respectively. As depicted in Figure 4, these two loops are embedded in the state transition diagram. A *strobe* signal is included in the state transition diagram for initiating the DCT. For clarity, several states are sometimes merged into a single oval in Figure 4. A symbolic state transition table may be directly derived from the state transition diagram. The fundamental difference between a Moore model and a Mealy model is analyzed and compared. In addition, the students are encouraged to apply multiple-code state assignment to prevent the controller from entering an undefined state.
- Implementation technology
  An FPGA board was chosen as the target technology for implementing the hardware due to its convenience and low cost [9].

So far a top-down approach has been applied to produce the global structure of the final design. Alternative global structures can be generated by exploring different schedules of the events. In the next section, we will discuss design tradeoffs for each building block.

4. ADDITIONAL CONSIDERATIONS

Many design alternatives are often available for a component such as a multiplier and multiplexer. Typical tradeoffs include the following:

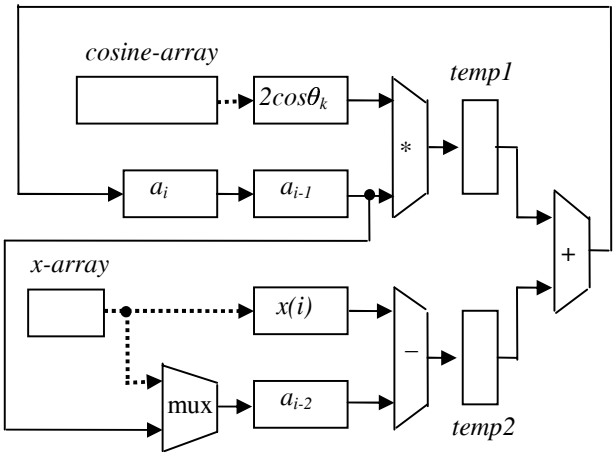

Figure 2: *a*-Coefficients Data Paths

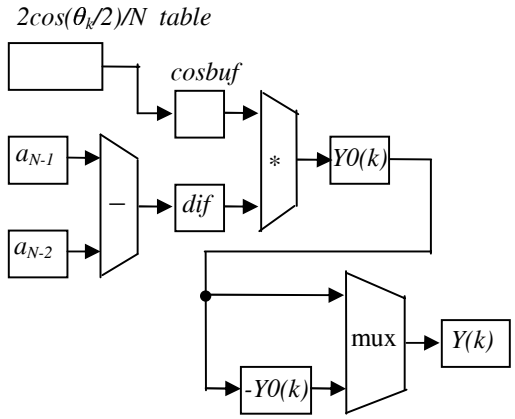

Figure 3: *Y*-Generator Data Paths

- Data representation
  The data representation is the major source of quantization and other residual errors. These errors are a major concern for digital signal processing [5]. An assignment for the students is to investigate the impact of the following issues.
  - The number of bits used to represent the input and output data, the coefficients, the constants, and intermediate buffers.
  - The position of binary point for each representation.
  - The ways of handling arithmetic operations.
    Typical selections include one's complement, two's complement as well as sign and magnitude. Generally speaking, two's complement is convenient for addition and subtraction while sign and magnitude is suitable for multiplication.
- Implementation options
  There are different ways of implementing some of the functional components. For example, a multiplier may be implemented as a combinational logic or a sequential circuit. An accumulator is required if a multiplier is implemented as a sequential circuit; the organization of the accumulator will then arise. In some cases, the students may study the implications of a normal implementation and a pipeline design. The constants may be realized as a memory block or directly tied to ground or power source. If a memory is used, there may be a tradeoff between single-port and dual-port

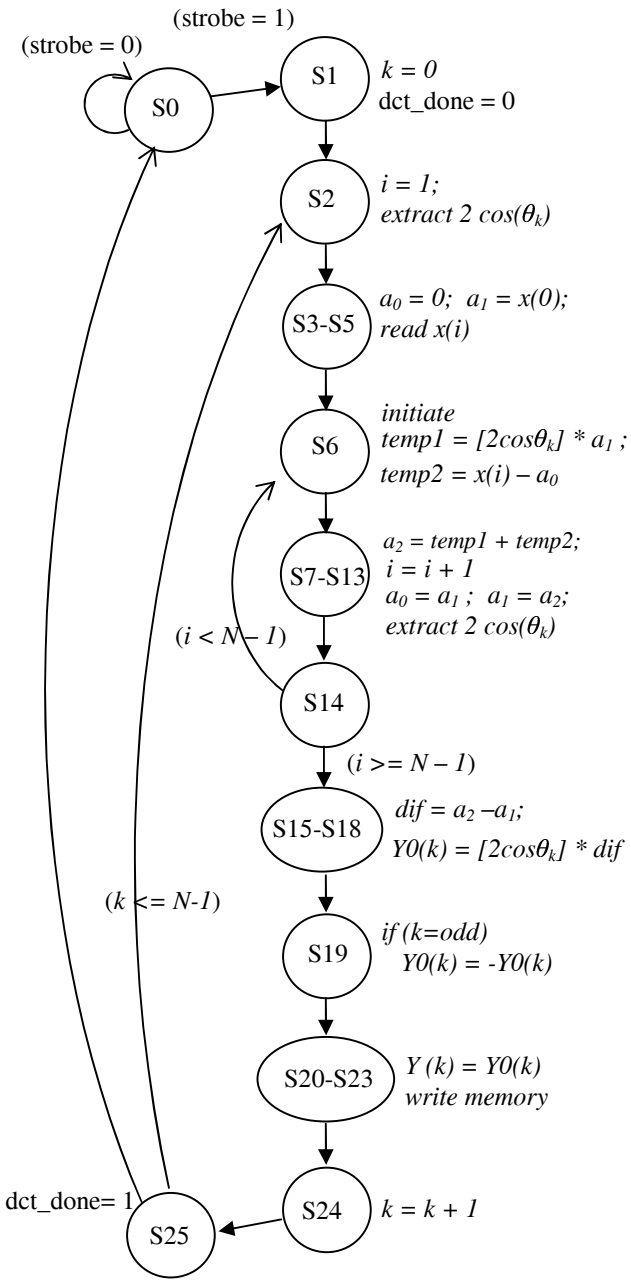

Figure 4: Control Flow for *a*-Coefficients and *Y*-Output Generators

memories. If an input to a logic gate is tied to ground or power source, one may address the effectiveness of using constant propagation for logic minimization.

- Clocking and event schedule
  The number of clock signals, the frequency of each clock signal, and the schedule of these clocks and other circuit events are all important issues in digital design. At Bucknell University, we are investigating efficient clocking and event schedules to endure seamless subsystem coordination and to improve system performance [8].
- Interface protocols
  Various interface methods including direct coupling, polling, and handshaking are addressed. The design of a task dispatcher and inter-module coordination schemes are discussed.
- VHDL coding
  In VHDL, a digital design may be described in several different styles including dataflow, behavioral, structural, and mixed styles. Students investigate these styles and their impact on the final implementation.
- Clock generation
  The methods of deriving clock signals with different frequencies and proper duty cycles as well as how to generate these clock signals are often overlooked by students. The students are reminded of the importance of these issues.
- Input, output, and inter-module interface
  DIP switches and push-buttons as well as light-emitting-device (LED) displays are readily available on the FPGA boards [9]. Students study various methods for effectively handling these input and output devices. Some methods focus on innovative approaches for capturing input data. For example, it is not possible to directly enter a three-digit decimal number on an FPGA board that we use. We instruct the students to design a circuit to display the ten decimal digits at low speed so that the displayed value can be observed. A push-button switch is then used to select the desired digit. The other alternative is to generate and display the ten decimal digits through a push-button switch. A second push-button switch is then used to select the desired number. These methods of capturing a decimal digit is repeated for the one's, the ten's and the hundred's digits. Also, several methods for eliminating or properly handling switch bouncing problems are hinted to the students.

Once the structures of all the building blocks are determined, the interface must be specified. The state transition diagram may also need to be refined. The students learn the essence of incremental and iterative designs.

5. TESTING AND DEBUGGING

Students are requested to perform functional simulation on their VHDL descriptions. Individual components are tested using the input and output devices available in an FPGA board before they are integrated into a larger system. The input and output devices on the FPGA boards are used to facilitate control flow testing. Checkpoints are inserted in the control flow to verify state transition and intermediate results produced by the data paths. This comprehensive testing approach based on control flow tracing is very effective for isolating design errors.

6. CONCLUSION

The DCT provides a rich set of tradeoffs for students to explore, including the selection of data width, the positioning of binary points, alternative methods for input data capture and output data display. Studying a DCT algorithm and using it as a vehicle for teaching digital system design, students are able to relate the theory and applications in digital signal processing with its hardware implementation. The methods of transforming an algorithm into a hardware implementation provide students with a concrete illustration in how a design can be produced. Because the approach is systematic, most students are able to produce a working design. Also, this methodology is applicable to the design of other computer algorithms, including the inverse Fourier transforms.

7. REFERENCES

[1] M. F. Aburdene, J. Zheng, and R. J. Kozick, "Computation of Discrete Cosine Transform Using Glenshaw's Recurrence Formula," IEEE Signal Processing Letters, Vol. 2, No. 8, pp. 155-156, August 1995.

[2] P. G. Fernandez, A. Garcia, J. Ramirez, L. Parrilla, and A. Lloris, "A New Implementation of the Discrete Cosine Transform in the Residue Number System," Proceedings of The Thirty-Third Asilomar Conference on Signals, Systems, and Computers, Vol. 2, pp. 1302-1306, 1999.

[3] S. F. Hsiao, W. R. Shiue, and J. M. Tseng, "Design and Implementation of a Novel Linear-Array DCT/IDCT Processor with Complexity of Order $\log_2 N$," IEE Proceedings on Vision, Image, and Signal Processing, Vol. 147, No. 5, pp. 400-408, October 2000.

[4] K. A. Kotteri, A. E. Bell, and J. E. Carletta, "Quantized FIR Filter Design: A Collaborative Project for Digital Signal Processing and Digital Design Courses," Proceedings of the 2004 ASEE Annual Conference & Exposition, Session 3232.

[5] A. V. Oppenheim and R. W. Schafer, "Discrete Time Signal Processing," Prentice-Hall, Inc., 1989.

[6] R. Scrofano, J. W. Jang, and V. K. Prasanna, "Energy-Efficient Discrete Cosine Transform on FPGAs," Proceedings of The 2003 International Conference on Engineering of Reconfigurable Systems and Algorithms, pp. 215-221.

[7] C. J. Tseng and D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-5, No. 3, pp. 379-395, July 1986.

[8] C. J. Tseng, "Clocking Schedule and Writing VHDL Programs for Synthesis," Proceedings of The 2004 ASEE Annual Conference & Exposition, Session 1532.

[9] Xess Corporation, XSA Board V1.1, V1.2 User Manual, Apex, North Carolina 27502, 2002.