Design of an Asynchronous Inverse Discrete Cosine Transform Circuit on an FPGA

Taiki Urakawa
University of Aizu
Aizuwakamatsu 965-8580, Japan
m5201140@u-aizu.ac.jp

Hiroshi Saito
University of Aizu
Aizuwakamatsu 965-8580, Japan
hiroshis@u-aizu.ac.jp

ABSTRACT

In this paper, we design a low power asynchronous Inverse Discrete Cosine Transform (IDCT) circuit on a Field Programmable Gate Array (FPGA). We synthesize the asynchronous IDCT circuit assigning the maximum delay constraints for data-paths. In the experiment, we evaluate the area, execution time, power consumption, and energy consumption of the designed asynchronous IDCT circuit. The power was reduced 24 % compared to the synchronous counterpart.

Categories and Subject Descriptors

B.6.0 [**LOGIC DESIGN**]: General

General Terms

Design

Keywords

Asynchronous circuits, FPGA, IDCT

1. INTRODUCTION

Recently, importance of image compression is increasing to save memories on portable devices. As portable devices are battery drive, low power image compression circuits are required for battery saving as mentioned in [1, 2, 3].

Inverse Discrete Cosine Transform (IDCT) is used in many multimedia applications such as JPEG [4] and MPEG [5]. IDCT is one of the heaviest processes in image compression. In IDCT, information in frequency domain is converted to information in time-space domain. As high performance and low power are required to IDCT, IDCT is usually realized as a specialized circuit.

Most of circuits including IDCT circuit are a synchronous circuit. Circuit components in synchronous circuits are controlled by global clock signals. Synchronization failure by clock skew and power consumption on the clock network will be significant in synchronous circuits when the semiconductor submicron technology is advanced more and more.

In asynchronous circuits, circuit components are controlled by local handshake signals. Due to the absence of global clock signals, there is no problem related to clock signals. In addition, asynchronous circuits are potentially low power consumption and low electromagnetic radiation compared to synchronous circuits. However, the design of asynchronous circuits is difficult more than the design of synchronous circuits due to the absence of global clock signals.

Recently, Field Programmable Gate Arrays (FPGAs) are well used in embedded systems. This is because FPGAs are long lifetime and low design cost. As designers can change circuit structure freely, FPGAs are adaptive for specification change and extension.

In [6], an asynchronous reconfigurable architecture was proposed for voice processing and image processing. This work used a special gate-level library for the design. It reported latency, throughput, and circuit area. However, the effectiveness is not well described because there is no comparative evaluation.

In this work, we design an asynchronous IDCT circuit. The asynchronous circuit is based on bundled-data implementation. After the design, we evaluate the designed asynchronous IDCT circuit comparing with the synchronous counterpart.

The rest of this paper is organized as follows. In section 2, we describe background used in this paper. In section 3, we describe the design of an asynchronous IDCT circuit. In section 4, we describe the experimental result. In section 5, we conclude this paper.

2. BACKGROUND

2.1 Asynchronous Circuits with Bundled-data Implementation

The bundled-data implementation is one of implementation methods of asynchronous circuits. It uses N+2 signals to represent N bit data. Additional 2 signals are request signal and acknowledge signal. In bundled-data implementations, the completion of operations are guaranteed by delay elements which are inserted on request signals. Therefore, the performance of bundled-data implementations depends on the delay of the control circuit with delay elements.

Figure 1 shows a circuit model of bundled-data implementation used in this paper. The right side of Fig. 1 is a data-path circuit and the left side is a control circuit.

The data-path circuit consists of registers (reg_k) ($0 \leq k \leq e$), functional units (fu), glue logics (g), and multiplexers (mux). They are the same as the ones used in synchronous circuits. In addition, the data-path circuit includes delay elements hd_k to satisfy hold constraints. Initially, hd_k is

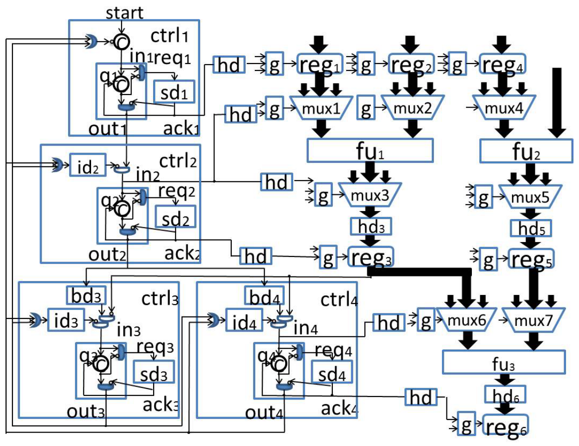

Figure 1: Asynchronous circuit with bundled-data implementation.

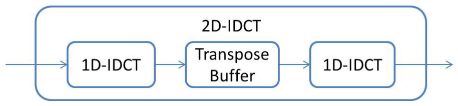

Figure 2: Structure of 2D-IDCT.

just a wire from the input to the output. Glue logics receive signals from the control circuits and generate control signals for functional units, multiplexers, and registers.

The control circuit consists of control modules $ctrl_i$ $(1 \leq i \leq n)$. A control module controls one state and consists of Q-module q_i, glue logic (g), three delay elements id_i, sd_i and bd_i, and C-element c_i. id_i is used to satisfy idle constraints. sd_i is used to satisfy setup constraints. bd_i is used to satisfy branch constraints. C-element c_i is a synchronization component of the input signals. The output of C-elements becomes 0 when all inputs are 0, it becomes 1 when all inputs are 1. Otherwise, the output is not changed.

The behaviors of bundled-data implementations are as follows. In this paper, $signal+$ represents a positive edge of $signal$, $signal-$ represents a negative edge of $signal$. This circuit model starts by $start+$ from outside. A control module $ctrl_i$ starts by $out_{i-1}+$ from $ctrl_{i-1}$. The control signals of functional units and multiplexers in the data-path circuit are generated by in_i+ through glue logics. q_i generates req_i+ by in_i+. req_i+ goes back to q_i through sd_i+ as ack_i+. q_i generates req_i- then goes back to q_i as ack_i-. After this, data are written into register by ack_i-. Finally, q_i generates out_i+ by ack_i- and moves the control to the next control module. After the last control module generates out_i+, all control modules generate in_i- and out_i-.

There are 4 types of timing constraints in asynchronous circuits with bundled-data implementation used in this paper. These are described in [7]. If some of constraints are not satisfied, we need to satisfy them adjusting delay elements sd_i, hd_k, bd_i, and id_i.

2.2 IDCT

Inverse Discrete Cosine Transform (IDCT) transforms signals from time-space domain into frequency domain. 2D-IDCT consists of two 1D-IDCT as Figure 2. The first 1D-IDCT processes row information and the last 1D-IDCT processes column information. We represent N-unit input data of the frequency domain as $(F_0, \cdots, F_{N-1})$, and N-unit output data of the space-time domain as $S(k)$ $(k = 0, \cdots, N-1)$. The formula of IDCT is represented as (1).

$$S(k) = \frac{1}{2}F_0 + \sum_{n=1}^{N-1} F_n \cos\left(\frac{\pi}{N} n \left(k + \frac{1}{2}\right)\right) \tag{1}$$

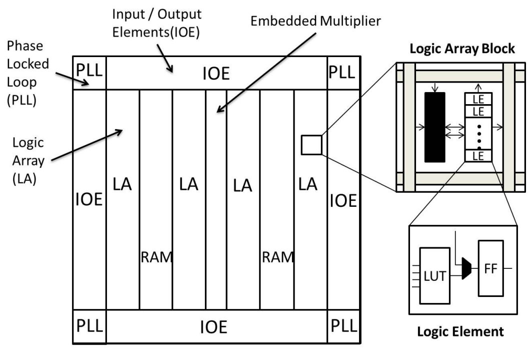

Figure 3: Structure of Altera Cyclone IV [8].

2.3 FPGA

Field Programmable Gate Array (FPGA) is a reconfigurable device where designer can change circuit structure freely. Therefore, FPGA is adaptive with specification change and expansion. The demands of FPGAs in embedded systems are increased because of its low design cost.

In this work, we implement an asynchronous IDCT circuit on an Altera FPGA. Figure 3 represents the structure of Altera Cyclone IV FPGA. It consists of Phase Locked Loops (PLLs), Input/output Elements (IOEs), Embedded Multipliers (EMs), Random Access Memories (RAMs), and Logic Arrays (LAs). PLLs are used for multiplication and division of clock signals, IOEs are used for inputs and outputs to outside. EMs are high performance multipliers and RAMs are memory blocks.

3. DESIGN OF AN ASYNCHRONOUS IDCT CIRCUIT

In this work, we design a low power asynchronous IDCT circuit on an FPGA. As power consumption depends on the execution time, we assign the maximum delay constraints for data-paths related to setup constraints. In this section, we describe the design of the circuit model, the generation of the maximum delay constraints, and the design flow.

3.1 Design of an Asynchronous IDCT Circuit Model

The data-path circuit of the asynchronous IDCT used in this work is the same as the data-path circuit of a synchronous IDCT. The control circuit is implemented by assigning a control module $ctrl_i$ to each state after making state transition diagram referring to the synchronous IDCT circuit.

Figure 4 shows the structure of the control module $ctrl_i$ for Altera Cyclone IV. It includes primitives DLATCH, LCELL, and AND2 of Altera Cyclone IV. Two dummy modules are inserted to assign wires as through points for path delay analysis. Dummy modules are just a wire. We insert two

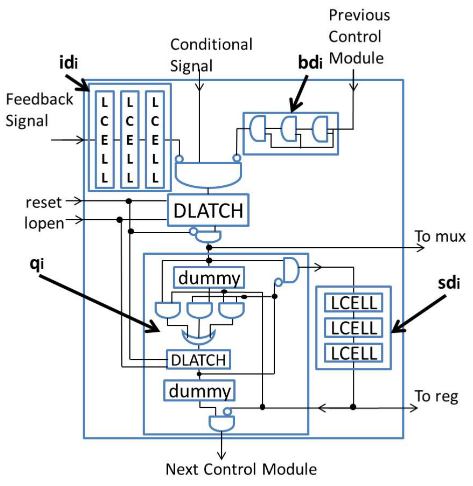

Figure 4: Structure of control module $ctrl_i$.

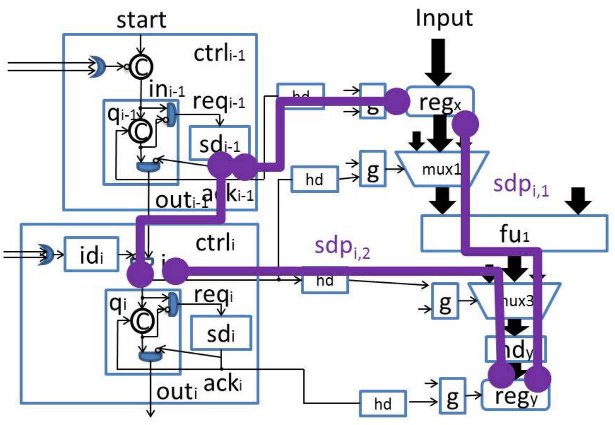

Figure 5: Data-paths to assign the maximum delay constraints.

```
set_max_delay
  -from ctrl0|sd0|out
  -to rr0|out[*]
  1.12
```

Figure 6: Example of set_max_delay command.

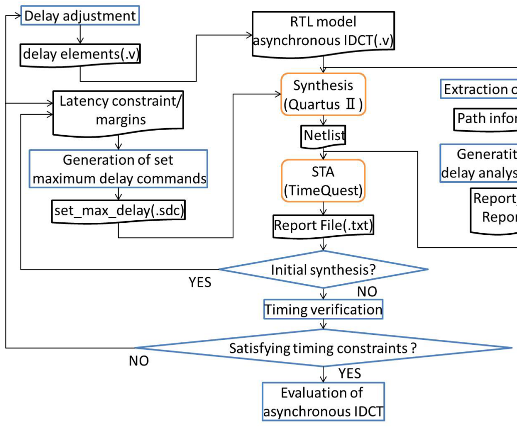
Delay adjustment
delay elements(.v)
RTL model asynchronous IDCT(.v)
Latency constraint/ margins
Synthesis (Quartus II)
Extraction of timing path
Path information file
Netlist
Generation of set maximum delay commands
Generatiton of path delay analysis commands
STA (TimeQuest)
set_max_delay(.sdc)
Report_timing/ Report_path
Report File(.txt)
Initial synthesis?
YES
NO
Timing verification
Satisfying timing constraints?
NO
YES
Evaluation of asynchronous IDCT

Figure 7: Design flow.

DLATCHes to analyze path delays correctly. The delay elements id_i and sd_i consist of LCELLs which work as buffers. The delay elements bd_i consist of AND primitives. We put "synthesis keep" attribute for the output pin of sd_i to analyze path delays correctly. Moreover, to avoid optimization for control modules, we assign "design partition" attribute for control modules. Finally, we represent the asynchronous IDCT circuit model using Verilog HDL.

3.2 Generation of the Maximum Delay Constraints and Setting of Margins

3.2.1 Generation of the Maximum Delay Constraints

We assign the maximum delay constraints to data-paths $sdp_{i,p}$ related to setup constraints. This is because we would like to design a bundled-data implementation which satisfies a given latency constraint.

Figure 5 represents two data-paths $sdp_{i,1}$ and $sdp_{i,2}$. A data-path $sdp_{i,p}$ is divided into 2 sub-paths. The former is from sd_{i-1} to source register reg_x or from sd_{i-1} to DLATCH before q_i. The latter is from reg_x to destination register reg_y or DLATCH to reg_y.

To decide the values of the maximum delay constraints, we use the calculation method in [7]. Figure 6 represents an example of the generated maximum delay constraints using "set_max_delay" command.

3.2.2 Setting of Margins

For data-path delays and control path delays, we setup margins. Uncertain effects to data-path delays after FPGA implementation may exist. Therefore, we setup a margin for data-paths. The margin is decided by a given latency constraint L and actual path delays on the target FPGA. If timing violations exist during simulation after the setup of the margin, we increase the margin to solve timing violations.

In bundled-data implementation, from setup constraints, the minimum delay of a control path $scp_{i,p}$ must be larger than the maximum delay of a data-path $sdp_{i,p}$. However, if the minimum delay of $scp_{i,p}$ is very large for the maximum delay of $sdp_{i,p}$, it may results in performance degradation. In such a case, we need to reduce primitives from delay elements. On the other hand, the reduction of primitives from delay elements increases the number of delay adjustments caused by timing violations. Therefore, we setup a margin for control paths. If the minimum delay of $scp_{i,p}$ overs the margin, we remove primitives from corresponding delay elements. The value of the margin is increased if the number of delay adjustments overs a threshold value

3.3 Desing Flow

Figure 7 shows a design flow used in this work. The inputs of this flow are a bundled-data implementation model represented by Verilog HDL (explained in section 3.1), a latency constraint L, and margins for data-path delays and control path delays.

First, we extract paths related to timing constraints from

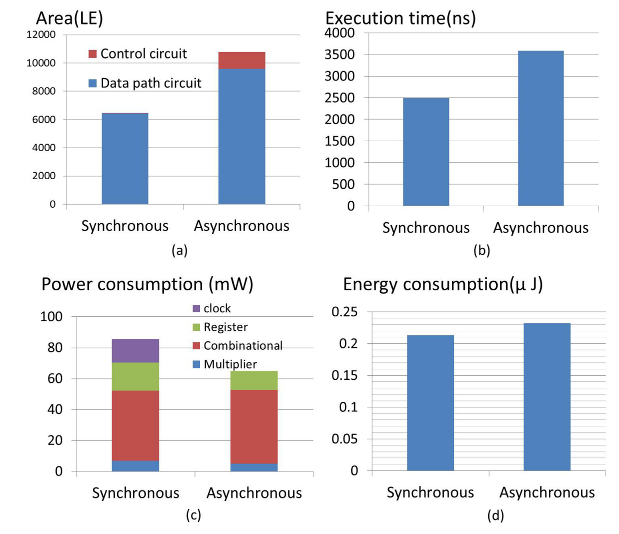

Figure 8: Experimental result: (a) area, (b) execution time, (c) dynamic power, and (d) energy consumption.

the bundled-data implementation model. The extracted information is represented in a path information file. From the path information file, we generate "report_timing" and "report_path" commands which are commands to analyze path delays using a timing analyzer. If the end point of a path is a register, we use "report_timing" to analyze the setup or hold time of the register. Otherwise, we use "report_path" command. Through the synthesis and static timing analysis using Altera Quartus II and TimeQuest, the flow is branched. If it is the initial synthesis, we generate the maximum delay constraints for data-paths related to setup constraints explained in section 3.2. If it is not the initial synthesis, we verify 4 types of timing constraints described in [7] manually. If all timing constraints are satisfied, we finish the design. Otherwise, we carry out delay adjustment to satisfy timing constraints. Synthesis is repeatedly carried out until all timing constraints are satisfied. If the number of timing constraints over a threshold value, we increase the margin for control path delays and repeat synthesis.

4. EXPERIMENTS

In the experiment, we evaluate the designed asynchronous IDCT circuit in terms of area, execution time, dynamic power consumption, and energy consumption comparing with a synchronous counterpart. We use Altera Quartus II ver.14.1 and ModelSim-Altera Starter Edition ver.10.3 for synthesis and simulation. The target device is Altera Cyclone IV (EP4CE115F29C7).

Initially, we explore the fastest synchronous IDCT circuit in clock frequency by changing clock cycle time. The clock cycle time of the fastest synchronous IDCT is 14 ns. The latency constraint for the asynchronous IDCT circuit is set to 308 ns. The margin for data-path delays is set to 1.4 ns (10% of clock cycle time) and the margin for control path delays is 5.6 ns (40% of clock cycle time).

Figure 8 (a) represents the area of the designed IDCT circuit in terms of logic elements reported by Quartus II. The area overhead of the asynchronous IDCT circuit is 33 % compared to the synchronous IDCT circuit. The overhead is caused by the area of control modules and delay elements. Figure 8 (b) represents the execution time when an arbitrary test input is given and simulated using ModelSim-Altera. The overhead of the execution in the asynchronous IDCT circuit is 44 % compared to the synchronous IDCT circuit. This overhead comes from the margin for control path delays. As we set a large enough value (5.6ns), more LCELLS are inserted to delay elements. Figure 8 .(c) represents the dynamic power consumption reported by PowerPlay Power Analyzer in Quartus II when the designed circuit and simulation result are given. We can reduce 24% of dynamic power consumption due to the reduction of the power consumption for clock network and register. Figure 8 .(d) represents the energy consumption obtained by the product of the dynamic power consumption and the execution time. It is increased 9% due to the increase of the execution time.

5. CONCLUSIONS

In this work, we designed a low power asynchronous IDCT circuit on an FPGA. The circuit was synthesized by using the maximum delay constraints for data-paths related to setup constraints. In the experiment, we confirmed that 24 % of power was reduced compared to the synchronous counterpart. However, the execution time and the energy consumption were increased due to the insertion of more LCELLs to delay elements.

As future work, we will optimize the execution time by assigning the maximum delay constraints for not only data-paths but also control paths.

6. REFERENCES

[1] J. Zhang, P. Chow, H. Liu, " FPGA Implementation of Low-Power and High-PSNR DCT/IDCT Architecture based on Adaptive Recoding CORDIC, " Proc. IEEE International Conference on Field Programmable Technology 2015, pp. 128-135, Dec. 2015.

[2] T. Xanthopoulos, A. Chandrakasan, " A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity, " Journal of Solid State Circuits, vol. 34, pp. 693-703, May 1999.

[3] S. R. Bhaisare, A. V. Gokhale, P. K. Dakhole, " CORDIC Architecture Based 2-D DCT and IDCT for Image Compression, " Proc. IEEE International Conference on Communication and Signal Processing 2015, pp. 1473-1477, Nov. 2015.

[4] G. K. Wallace, " The JPEG Still Picture Compression Standard, " IEEE Transactions on Consumer Electronics, vol. 38, no. 1, pp. xviii-xxxiv, Feb. 1992.

[5] D. L. Gall, " MPEG: a video compression standard for multimedia applications, " Communications of the ACM - Special issue on digital multimedia systems, vol. 34, no. 4, pp. 46-58, Apr. 1991.

[6] A. Rettberg, B. Kleinjohann, " A Fast Asynchronous Re-configurable Architecture for Multimedia Applications, " Proc. 14th Symposium on Integrated Circuits and Systems Design 2001, pp. 150-155, Sep. 2001.

[7] K. Takizawa, S. Hosaka, H. Saito, " A Design Support Tool Set for Asynchronous Circuits with Bundled-data Implementation on FPGAs, " Proc. IEEE 24th International Conference on Field Programmable Logic and Applications (FPL), pp. 1-4, Sep. 2014.

[8] Altera Cyclone IV, [https://www.altera.com/products/fpga/cyclone-series/cyclone-iv/features.html].