Project Space Exploration on the 2-D DCT Architecture of a JPEG Compressor Directed to FPGA Implementation

Roger Endrigo Carvalho Porto, Luciano Volcan Agostini
Group of Architectures and Integrated Circuits
Universidade Federal de Pelotas – UFPel, DMEC, Pelotas, Rio Grande do Sul, Brazil
Campus Universitário, s/nº – Caixa Postal 354 – CEP 96010-900
{rogerecp, agostini}@ufpel.edu.br

Abstract

This paper presents a project space exploration on the baseline JPEG compressor proposed and implemented in previous works. This exploration took as basis the substitution of the operators used in the 2-D DCT calculation architecture of the compressor and the consequent evaluation of impact in terms of performance and resources utilization. This substitution was made with main focus in the carry lookahead, hierarchical carry lookahead and carry select architectures, with the objective to increase the JPEG compressor performance. As the compressor architecture was designed in an hierarchical mode the operators substitution was an activity quite simple, because it has not involved the other hierarchy levels. The operators were described in VHDL, synthesized and validated. They were inserted in the 2-D DCT architecture for synthesis in the whole module. The 2-D DCT was synthesized for an Altera FPGA. With this project space exploration, the highest performance obtained for the 2-D DCT was 23% higher than the original, using 11% more logic cells.

1. Introduction

Discrete Cosine Transform (DCT) is a mathematical tool that has a lot of electronic applications, from audio filters to video compression hardware. DCT transforms the information from the time or space domains to the frequency domain, such that other tools and transmission media can be run or used more efficiently to reach application goals: compact representation, fast transmission, memory savings, and so on.

The JPEG image compression standard [1, 9] was developed by Joint Photographic Expert Group [10]. The JPEG compression principle is the use of controllable losses to reach high compression rates. In this context, the information is transformed to the frequency domain through DCT. Since neighbor pixels in an image have high likelihood of showing small variations in color, the DCT output will group the higher amplitudes in the lower spatial frequencies [11]. Then, the higher spatial frequencies can be discarded, generating a high compression rate and a small perceptible loss in the image quality.

The JPEG compression of gray scale images can be divided into three main steps, as shown in Fig. 1: 2-D DCT, quantization and entropy coding.

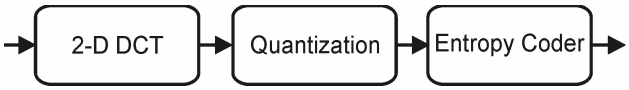

Fig. 1 – JPEG baseline compression

In the JPEG compressor, the 2-D DCT is the most critical module to be implemented because of its high algorithm complexity. In the architectural level this critical point is responsibility of the sum operations on wide inputs and of the consequent delay generated by the carry propagation in the ripple carry architecture used in the adders [2].

Thus, the main objective of this work was to increase the JPEG compressor performance through the substitution of the architectures used in the adders, with main focus in the carry lookahead, hierarchical carry lookahead and carry select architectures [4, 5, 6]. Were evaluated, too, the impacts in terms of performance and resources utilization.

The paper sections present the 2-D DCT and 1-D DCT architectures, the alternative architectures of operators used in this work, the description and validation of the operators, the synthesis results and conclusions.

2. Two dimensional DCT architecture

The 2-D DCT architecture used in the JPEG compressor is generically presented in Fig. 2. This architecture was designed to reach a high operating frequency and to allow the use of pipeline techniques and

1530-1591/04 $20.00 (c) 2004 IEEE

is based on the architecture proposed in [12] with some modifications. Thus, the architecture was divided into two 1-D DCT architectures and one transpose buffer. The two 1-D DCT architectures are similar but the bit widths at each pipeline stage are different. The 1-D DCT architectures are organized in a six stage pipeline, one stage for each algorithm step [2]. The transpose buffer operates like a temporal barrier between the first and the second 1-D DCT, allowing the use of a 2-D DCT global pipeline.

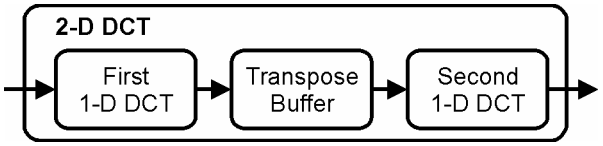

Fig. 2 – Generic 2-D DCT architecture

The 2-D DCT inputs in our design are matrixes of 8x8 elements eight-bit wide each. The first 1-D DCT receives and processes these matrixes in a row-wise order. The transpose buffer receives the row-wise results and gives the column-wise inputs to the second 1-D DCT architecture. The second architecture processes the column-wise data and gives a column-wise data output.

2.1. One dimensional DCT architecture

The 1-D DCT architecture proposed in [12] and used in this paper is presented in the Fig. 3

Considering the 1-D DCT algorithm steps, the use of a pipelined architecture between these steps becomes natural. Since the algorithm has six steps, the pipeline will have six stages, where five perform additions/subtractions and one performs multiplications.

The five adders in the original DCT architecture are ripple-carry and the multiplier is based on shift-add operations. This architecture uses a single arithmetic unit in each stage to perform all the necessary operations at that stage.

Then the unit inputs are connected by multiplexers to select which value must be used in each clock cycle. The multiplexers' controls are generated following the algorithm order. Temporal barriers to allow the pipeline design are obtained with the use of ping-pong registers. The control block generates the signals to control the pipeline fill-up and emptying through an external signal that indicates if the input values are valid values for the image.

The adders' architecture used in the original 1-D DCT is ripple carry [4].

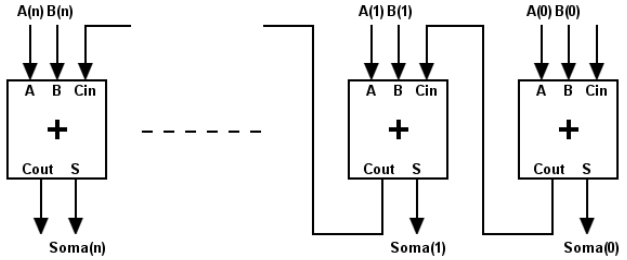

Fig. 4 – Block diagram of a ripple carry adder

This architecture of adders is quite simple and widely spread. Its main advantage is the reduced occupied area. Even so, there is the disadvantage of the low performance provoked by the slow carry propagation. This slow carry propagation is that defines the critical path of the architecture of 1-D DCT calculation and, for consequence, of own 2-D DCT architecture. Fig. 4 presents the architecture in blocks of a ripple carry adder.

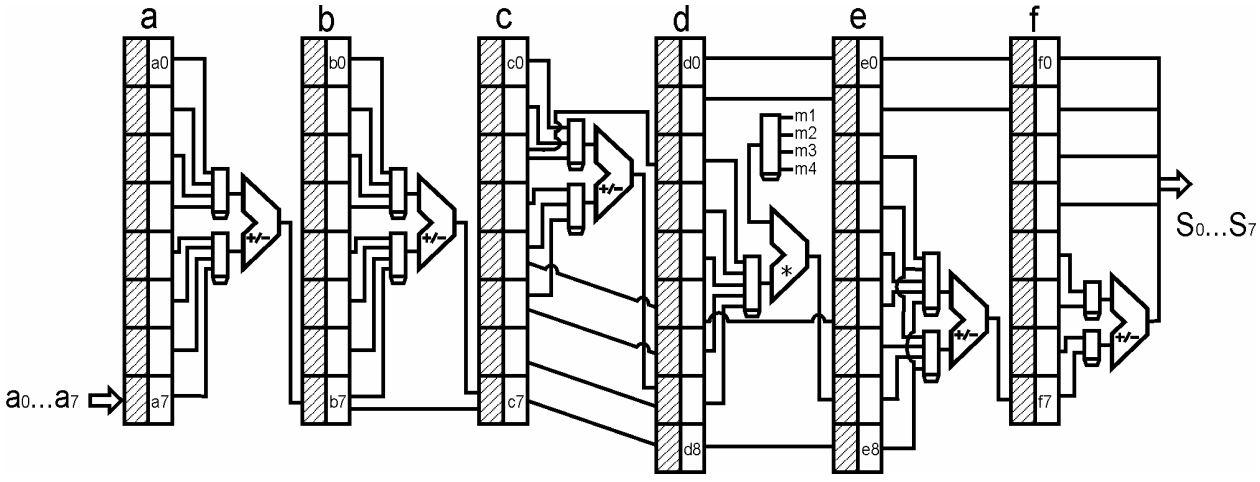

Fig. 3 – One dimensional DCT architecture

Being C_{i-1} the stage carry in, A_i and B_i the adder inputs, C_i the stage carry out and S_i the resulting sum, we can write the equation that determines the sum as:

$$S_i = A_i \cdot B_i \cdot C_{i-1} + A_i \cdot \overline{B_i} \cdot \overline{C_{i-1}} + \overline{A_i} \cdot \overline{B_i} \cdot C_{i-1} + \overline{A_i} \cdot B_i \cdot \overline{C_{i-1}}$$

This can be factored in the following way:

$$S_i = C_{i-1} \cdot \left(A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i} \right) + \overline{C_{i-1}} \cdot \left(A_i \cdot \overline{B_i} + \overline{A_i} \cdot B_i \right)$$
$$= A_i \oplus B_i \oplus C_{i-1}$$

The equation that determines the carry can be written as:

$$C_i = A_i \cdot B_i + A_i \cdot C_{i-1} + B_i \cdot C_{i-1}$$

This, also, can be factored in the following way:

$$C_i = A_i \cdot B_i + C_{i-1} \cdot \left(A_i + B_i \right)$$

By the fact that each stage needs the carry produced in the previous stage to make the sum, the total time delay is linearly proportional to the length *n* of the adder.

In [2] a small logic was developed (that can be seen in the Tab. 1) involving the input B, the adder carry in and the operation control signal *AddSub*. With this logic a same arithmetic operator can operate in addition mode or in subtraction mode.

Tab. 1 – Adders operation mode control logic

AddSub	Carry In	Input B	Operation
0	0	B	A + B
1	1	~B	A + (~B + 1) or A – B

Besides, it was possible to determine the maxima values generated by each operator and, like this, to identify which adders would not generate, in any hypothesis, a significant carry out. This way, it was possible to use the minimum bit width in each stage and, in consequence, the minimum of resources.

In the Tab. 2 are presented, for each stage of the two architectures of 1-D DCT, the bit width of each adder, which adders have the generation of carry out and which adders can operate in the subtraction way. In this table, the stage of the multiplier (stage 4) presents the three adders used by its architecture.

3. Some alternative adder architectures

This section will present the alternative adder architectures that were used in this work.

Tab. 2 – Adders used in each stage of the first and the second 1-D DCT

Stage	1st 1-D DCT			2nd 1-D DCT		
	Bits	Carry out	Add / Sub	Bits	Carry out	Add / Sub
1	8	yes	yes	12	yes	yes
2	9	yes	yes	13	yes	yes
3	10	yes	yes	14	yes	yes
4	16	no	no	20	no	no
	16	yes	no	20	yes	no
	20	no	no	24	no	no
5	11	yes	yes	15	no	yes
6	12	no	yes	15	no	yes

3.1. Carry Lookahead Adders

This kind of adder architecture use a technique called carry lookahead that have as purpose to speed up the carry propagation in an adder [6]. This technique bases on examining all the input stages of an adder and, simultaneously, to produce the appropriate carries for each one of these stages, that is to say, all the carries are calculated at the same time, in parallel.

Being A_i and B_i the inputs in a *n*-bit adder, C_{i-1} the carry in of the stage; C_{-1} the carry into the least significant position and S_i and C_i the sum and carry out of the stage; we can define two auxiliary functions:

$$G_i = A_i \cdot B_i$$

$$P_i = A_i \oplus B_i$$

The carry generate function (G_i) reflects the condition that a carry is originated at the *i*-th stage. The function P_i, called carry propagate, is true when the *i*-th stage will pass the incoming carry (C_{i-1}) to the next higher stage. Substituting P_i e G_i into ripple carry equations, we obtain

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$
$$= P_i \oplus C_{i-1}$$

$$C_i = A_i \cdot B_i + \left(A_i \oplus B_i \right) \cdot C_{i-1}$$
$$= G_i + P_i \cdot C_{i-1}$$

Thus, the sum operation, being used carry lookahead, is composed by three stages, as it can be observed in Fig. 5.

First, all the signals generate and propagate are generated simultaneously. After, and also in a simultaneous way, the carries are generated. Finally, having all the carries and signals G_i, the sum is generated.

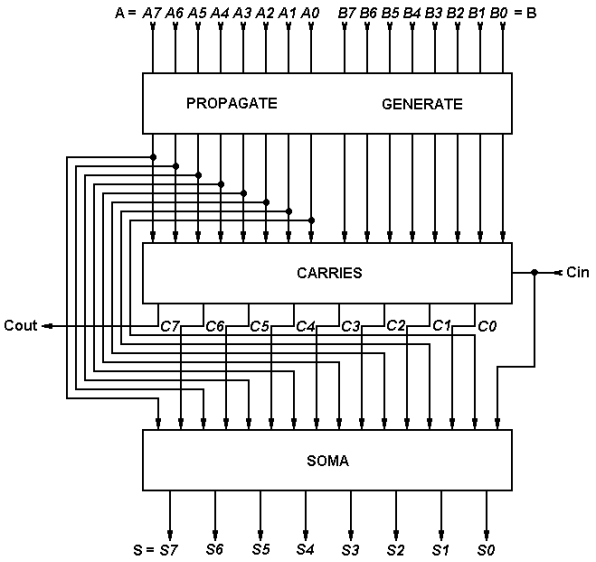

Fig. 5 – Block diagram of an 8-bit carry lookahead adder

3.2. Hierarchical Carry Lookahead Adders

Another architecture to be analyzed is the hierarchical carry lookahead. The purpose of this approach, besides to increase the speed of carry propagation, is to decrease the complexity of the produced equations when carry lookahead is used in an adder. As consequence, is had an hierarchical adder equivalent to a carry lookahead but reaching a better performance than this last one.

In this architecture type a control block is used to group *m* adders of *n*-bits width, that we will call adder blocks, forming a *m* x *n*-bits adder. The behavior of the adder blocks is common to the behavior of the carry lookahead adders except for the generation of the signals *P** (block carry propagate) and *G** (block carry generate). These signals can be written as:

$$P^* = P_{n-1} \cdot P_{n-2} \cdot \ldots \cdot P_1 \cdot P_0$$

$$G^* = G_{n-1} + P_{n-1} \cdot G_{n-2} + P_{n-1} \cdot P_{n-2} \cdot G_{n-3} + \ldots + P_{n-1} \cdot \ldots \cdot P_1 \cdot G_0$$

The control block, for its time, generates the appropriate carries out. These carries out are used as carries in in the subsequent adders blocks and can be written as:

$$Cout = G^* + P^* \cdot Cin$$

where P^* and G^* are produced in the actual stage and *Cin* in the previous stage.

For clarity, Fig. 6 illustrates an example of this kind of adder.

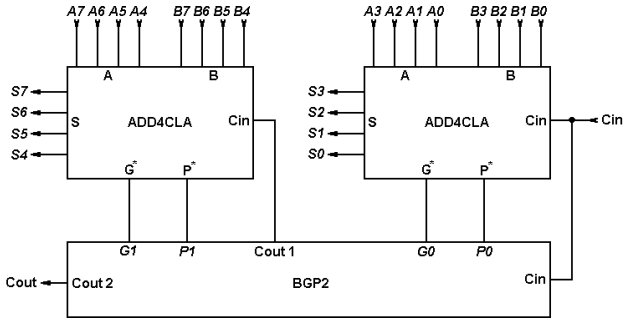

Fig. 6 – Block diagram of an 8-bit hierarchical carry lookahead

3.3. Carry Select Adders

An additional approach to speed up the carry propagation are the carry select adders [4]. Usually, an carry select adder is divided in two adder sessions. Ripple carry propagation is assumed within the adder sections. Each adder section is in duplicate, one with a carry and one without carry into the lowest-order bit in the section [6]. A multiplexer selects the appropriate sum in each section. The basic scheme of an 8-bit carry select adder is shown in the Fig. 7.

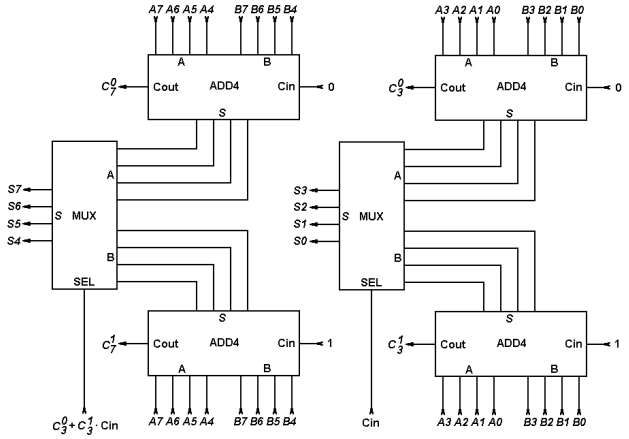

Fig. 7 – Block diagram of an 8-bit carry select adder

4. VHDL adders description

The description of the operators designed in this work was made in VHDL, using the Max Plus II environment [8]. Were designed 18 descriptions files for the carry lookahead architecture, 38 for the hierarchical carry

lookahead architecture and 19 for the carry select architecture, totaling 75 description files for the three presented alternative architectures.

The amount of descriptions implemented for each architectural alternative differs of the original amount (15 descriptions). This is due to the need of auxiliary files so that the specific characteristics of each architecture were respected. In the case of the hierarchical carry lookahead architecture two alternatives were designed for each operator.

In the implementation of the hierarchical carry lookahead and carry select architectures the advantage of the hierarchization was used through the reuse of the code of another descriptions. In other words, blocks common to several operators were described in a separate way so that they could be referenced as components in these adders. Herewith the code of the descriptions was, besides shorter, clearer. The Fig. 8 presents, as example, the hierarchy of the descriptions used for a 8-bit hierarchical carry lookahead adder/subtractor. This operator is formed by three components: two adders grouped by a control block.

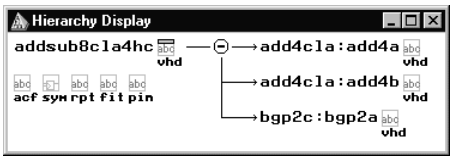

Fig. 8 – Example of description hierarchy of one 8-bit hierarchical carry lookahead adder / subtractor

Later on, the original operators of 2-D DCT (ripple carry adders) were substituted by the new implemented operators so that it was made the synthesis and evaluation of results.

5. Synthesis results

Initially, all the described operators were synthesized separately. After that, they were synthesized inside of the first and second 1-D DCT modules. Finally, 2-D DCT was synthesized with the new described operators.

The results of the synthesis for Altera ACEX1K FPGAs [8] are presented in the following tables. The synthesis results for the first 1-D DCT, architecture by architecture, can be compared in the Tab. 3. Tab. 4 presents the same comparison type but for second 1-D DCT. The general results, that is to say, for the 2-D DCT module, are shown in tab.5.

Tab. 5 shows the general results, that is to say, a comparison of the 2-D DCT synthesis results for an Altera FPGA of ACEX1K family (EP1K100QC208-1) being used the three designed alternative architectures.

For the carry lookahead architecture the 2-D DCT used 4192 logic cells and reached 22,67 MHz as maximum operating frequency. In the case of hierarchical carry lookahead, were used 4343 logic cells, reaching a maximum frequency of 24,03 MHz. Finally, for the carry select architecture were used 4621 logic cells to obtain a maximum frequency of 27,10 MHz.

Tab. 3 - Comparative synthesis results for the first 1-D DCT

1st 1-D DCT	RCA	CLA	H. CLA	CSA
Logic Cells	1660	1664	1734	1842
Period (ns)	37,8	35,3	34,8	34,3
Frequency. (MHz)	26,45	28,32	28,73	29,15
Frequency Gain (%)	-	7,08	8,62	10,21
Resources Loss (%)	-	0,24	4,46	10,96

Tab. 4 - Comparative synthesis results for the second 1-D DCT

2nd 1-D DCT	RCA	CLA	H. CLA	CSA
Logic Cells	2241	2243	2331	2490
Period. (ns)	41,1	39,6	40,8	40,1
Frequency (MHz)	24,33	25,25	24,5	24,93
Frequency Gain (%)	-	3,78	0,7	2,47
Resources Loss (%)	-	0,09	4,02	11,11

Tab. 5 - Comparative synthesis results for the 2-D DCT

2-D DCT	RCA	CLA	H. CLA	CSA
Logic Cells	4181	4192	4343	4621
Period (ns)	45,7	44,1	41,6	36,9
Frequency (MHz)	21,88	22,67	24,03	27,10
Frequency Gain (%)	-	3,61	9,83	23,86
Resources Loss (%)	-	0,26	3,87	10,52

6. Operators validation

The operators' validation considered the results of synthesis and of the simulation. The simulation of the operators was made with the aid of the Max Plus II waveform editor (Fig. 9).

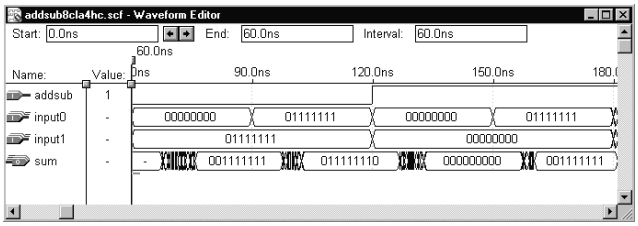

Fig. 9 - The Max Plus II waveform editor

In the simulation process the correct operation of the operators was evaluated being observed the behavior of the output in function of the input values.

Not only the operators were simulated but also all the implemented descriptions including the auxiliary files.

7. Conclusions

This paper presented the design of adder architectures for use in JPEG compression, more specifically for use in the 2-D DCT of the JPEG compressor.

Were designed 75 VHDL descriptions for the three architectural alternatives used in the implementation of the operators. The results of the synthesis were presented too.

The maxima DCT 2-D operating frequencies, obtained as result for each one of the implemented alternative architectures, were all larger ones than to original maximum operating frequency obtained, for example, a gain in performance around 24% to the carry select architecture.

Being considered that the main objective of the work were increase the performance of the JPEG compressor, the obtained results were considered satisfactory.

8. References

[1] The International Telegraph and Telephone Consultative Committee (CCITT). "Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines". Rec. T.81, 1992.

[2] L. Agostini. *Design of Architectures for JPEG Image Compression (Portuguese)*. Master Dissertation – Federal University of Rio Grande do Sul. Informatics Institute. Pos-Graduation in Computer Science Program, Porto Alegre, Brazil-RS, 2002. 143p.

[3] L. Agostini, I. Silva, S. Bampi . "Pipelined Fast 2-D DCT Architecture for JPEG Image Compression". In: SBCCI2001 - XIV Symposium on Integrated Circuits and System Design, 2001, Pirinópolis - GO - Brazil. p. 226-231.

[4] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design.- Second Edition*, Addison-Wesley, USA, 1995.

[5] J. Rabaey. *Digital Integrated Circuits : A Design Perspective*, Prentice Hall, USA, 1996.

[6] K. Hwang. *Computer Arithmetic: Principles, Architecture and Design*, John Wiley & Sons, USA, 1979.

[7] R. Airan, J. Berge and V. Olive. *Circuit Synthesis with VHDL*, USA, 1994.

[8] "Altera: The Programmable Solutions Company". San Jose, Altera Corporation, 2002. <http://www.altera.com>.

[9] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, USA, 1992.

[10] "Home site of the JPEG and JBIG committees" <http://www.jpeg.org>

[11] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards Algorithms and Architectures – Second Edition*, Kluwer Academic Publishers, USA, 1999.

[12] M. Kovac and N. Ranganathan. "JAGAR: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard". Proceedings of the IEEE, vol. 83, n°. 2, 1995, pp. 247-258.