Journal of Advanced Research (2016) 7, 909–922

Cairo University

Journal of Advanced Research

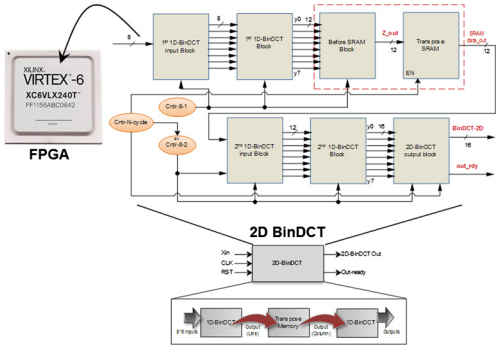
ORIGINAL ARTICLE

Efficient BinDCT hardware architecture exploration and implementation on FPGA

Abdessalem Ben Abdelali [a,b,*], **Ichraf Chatti** [a], **Marwa Hannachi** [a,c], **Abdellatif Mtibaa** [a]

[a] *Laboratory of Electronics and Microelectronics, University of Monastir, Tunisia*
[b] *High Institute of Informatics and Mathematics of Monastir, Monastir, Tunisia*
[c] *Institut Jean Lamour (IJL) UMR7198, University of Lorraine, Vandoeuvre Les Nancy, France*

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 2 June 2016
Received in revised form 25 August 2016
Accepted 5 September 2016
Available online 14 September 2016

ABSTRACT

This paper presents a hardware module design for the forward Binary Discrete Cosine Transform (BinDCT) and its implementation on a field programmable gate array device. Different architectures of the BinDCT module were explored to ensure the maximum efficiency. The elaboration of these architectures included architectural design, timing and pipeline analysis, hardware description language modeling, design synthesis, and implementation. The developed BinDCT hardware module presents a high efficiency in terms of operating frequency and

* Corresponding author.
E-mail address: Abdessalem.BenAbdelali@enim.rnu.tn (A. Ben Abdelali).
Peer review under responsibility of Cairo University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jare.2016.09.002

Keywords:
Binary discrete cosine transform
Discrete cosine transform approximation
Very large scale integration architectures
Hardware implementation
Design exploration
Field programmable gate array

hardware resources, which has made it suitable for the most recent video standards with high image resolution and refresh frequency. Additionally, the high hardware efficiency of the BinDCT would make it a very good candidate for time and resource-constrained applications. By comparison with several recent implementations of discrete cosine transform approximations, it has been shown that the proposed hardware BinDCT module presents the best performances.

Introduction

The Discrete Cosine Transform (DCT) is widely used in video and image processing. However, its direct implementation is not very efficient because it is a floating-point transform, and it is characterized by complex loops. In fact, floating-point algorithms are slow in software and require more silicon in hardware implementation [1–3]. Nevertheless, most video and image processing applications are subject to hard real time constraints, and the DCT must be calculated in a very short time. To decrease the complexity of this transform, several integer-friendly approximations of the DCT have been proposed [4,5]. The Binary Discrete Cosine Transform (BinDCT) [6] represents one of the most known DCT approximations. It works only with integer numbers, and it uses just sums and shifts. No multiplications are needed, hence reducing the execution time and the hardware resources.

In embedded system for video and image processing, the main objective is generally to meet the real-time constrains while assuring the maximum hardware efficiency. Indeed, high speed and energy efficiency are very important factors for numerous applications. In the literature several recent works [7–14] have addressed the hardware implementation of the DCT for real time applications and the decrease in computational complexity of this transformer. Different DCT approximations have been proposed to reduce the required number of addition and subtraction operations with no multiplication. In the suggested works, the Very Large Scale Integration (VLSI) hardware efficiency is sometimes promoted on accuracy. This can be tolerable because of the limited perception of human visualization that allows numerical approximations of the algorithm.

With its high performance and low complexity, the BinDCT hardware accelerator is an excellent candidate for real-time DCT-based image and video processing applications [15]. However, in the literature only some works have been interested in the hardware implementation of the BinDCT [16–20]. Moreover, no complete architectural exploration has been performed, and the majority of the existing works are relatively old and no hardware implementation employing the efficiency of novel reconfigurable systems technology has been put forward. They present limited performances that do not permit supporting actual high video standards.

In this paper, a design exploration of the 2D-BinDCT for its efficient hardware implementation on a Field Programmable Gate Array (FPGA) device is proposed. Also, a comparison with the hardware implementation of several existing DCT approximations is performed to demonstrate the efficiency of the suggested BinDCT implementation. The purpose here is to put forward an efficient hardware module of the BinDCT that can be employed in highly constrained systems.

The rest of the paper is organized as follows. The methodology section provides a review of the BinDCT algorithm and its structure, the architecture of the different BinDCT stages, and the global hardware architecture of the 2D transform. In the results and disscussion section, the implementation details and operations of the proposed 2D-BinDCT architectures are discussed. A comparison with the hardware implementations state of the art of the DCT approximations is also made in the latter section.

Methodology

Structure of BinDCT transform

The DCT is characterized by complex loops, cosine functions, and floating-point multiplications, which increase the computation time. Table 1 [6,15] shows the most known DCT algorithms. Chen's algorithm uses 16 multiplications (MUL) and 26 additions (ADD), which can be respectively optimized to 13 MUL and 29 ADD. Loffler's algorithm is the most optimized one. It utilizes only 11 MUL and 29 ADD. Loffler's and Chen's algorithms give the same performance as the classical DCT, and they are less complex and faster, but they still need floating point multiplication. The BinDCT is a fast DCT approximation [6], which uses only adders and shifters to reduce the computational complexity.

In our design, the lifting structure of the BinDCT family based on the Chen's factorization (BinDCT type C) was utilized. Fig. 1 illustrates the flowchart of the forward BinDCT. Depending on the lifting parameters, several BinDCT configurations exist for Chen's factorization: BinDCT-C1 to the BinDCT-C9 [4]. These configurations present a different number of arithmetic operations. BinDCT-C1 is the most accurate approximation among the other configurations, but it presents a higher computational cost. The BinDCT-C9 has the least accuracy and computation time. Each BinDCT configuration has different P and U value approximations, and consequently a different number of additions and shift operations.

In this paper, the C7 configuration is applied. Besides the medium complexity of this configuration, as mentioned by Liang and Tran [4], it actually presented a satisfactory biorthogonal coding gain, which was very close to that of

Table 1 Complexity of different DCT algorithms.

DCT operation	Chen	Wang	Lee	Vetterli	Houn	Loeffler	BinDCT C5
Mul	16	13	12	12	12	11	0
Add	26	29	29	29	29	29	36
Shifts	–	–	–	–	–	–	17

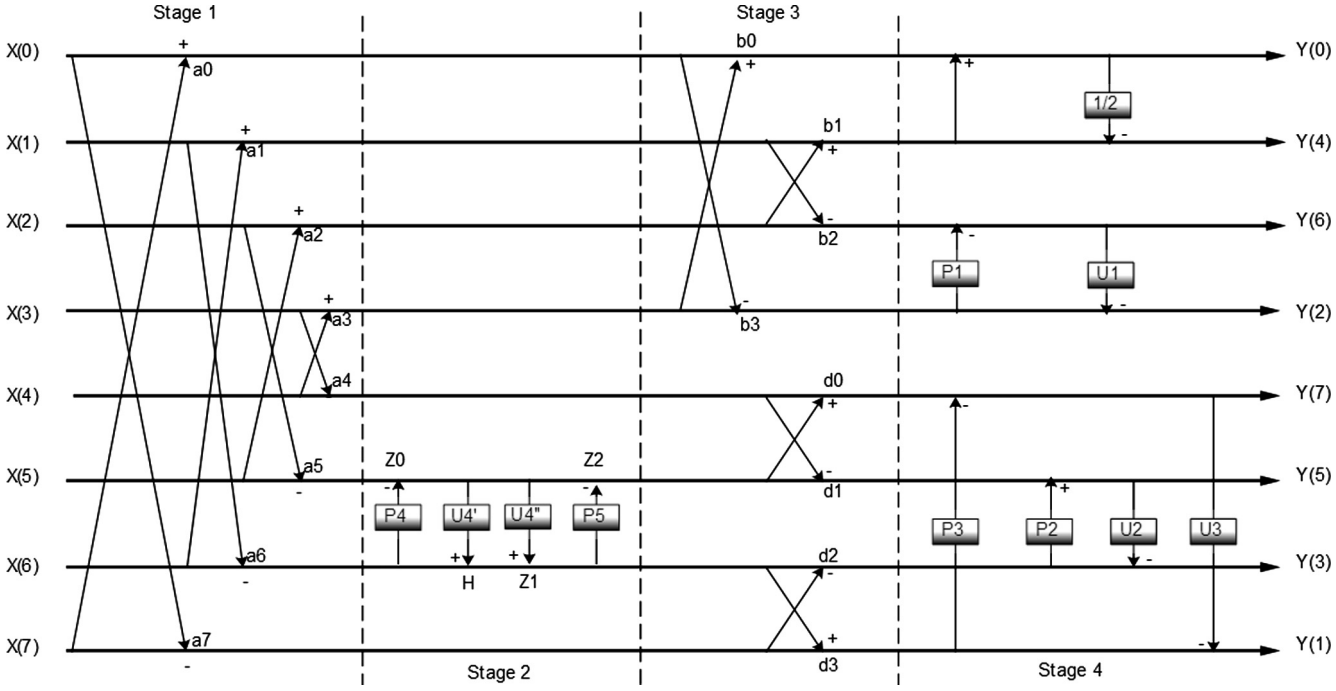


Fig. 1 Diagram of the BinDCT.

the DCT, and a relatively small mean square error (MSE) with true DCT outputs. In addition, a comparison of the Peak Signal-to-Nose Ratio (PSNR) results of the reconstructed Lena image with the C7 configuration of the BinDCT, the floating version and the fast integer version of the source code from the Independent JPEG Group (IJG) was performed. It was demonstrated that the performances of the employed BinDCT were close to the floating DCT implementation. In particular, when the quality factor is below 90, the difference between the C7 configuration of the BinDCT and the floating DCT will be below 0.5 dB. Besides, when the quality factor is above 90, the performance of the BinDCT will be much better than the fast integer version.

The C7 BinDCT configuration coefficients are the following:

$$P_1 = 1/2;\ U_1 = 1/2;\ P2 = 1;\ U_2 = 1/2;\ P_3 = 1/4;$$
$$U3 = 1/4;\ P_4 = 1/2;\ U_4 = 1/4 = U'_4 + U''_4 = 1/2 + 1/4,\ P5 = 1/2$$

In Fig. 1, the U_4 parameter, which is used to calculate the Z1 output of stage 2 ($Z_1 = U_4 \times Z_0 + a_6$), is transformed to an addition of U'_4 and U''_4 coefficients, and an intermediate sum is introduced ($H = U'_4 \times Z_0 + a_6$). The addition and shift operations corresponding to the Pi and Ui coefficients are defined as follows:

- ½×: shift right by one bit
- ¼×: shift right by 2 bits
- ¾× = ½× + ¼×: one addition and two shifts.

The 2D-BinDCT module is composed of two 1D-BinDCT modules and a transpose memory block. The first 1D-BinDCT block is applied to 8 × 8 matrix input data, and the second one is applied to the transpose matrix of the obtained 1D-BinDCT (8 × 8 matrix). The transposition is ensured by the transpose memory block.

The architecture of a 1D-BinDCT hardware module is composed of an input block with 8 registers and 4 hardware blocks corresponding to the four 1D-BinDCT stages. 4 additions and 4 subtractions are needed for stage 1, 2 additions, and 2 subtractions for stage 2, 4 additions and 4 subtractions for stage 3, and finally 2 additions and 6 subtractions for stage 4.

For the first 1D-BinDCT block, the four stages are called stages 1, 2, 3 and 4, while for the second one, they are called stages 11, 22, 33 and 44. Stages 1, 2, 3, 4, 11, 22, 33 and 44 have input data widths of 8, 9, 10, 11, 12, 13, 14 and 15 bits, and output data widths of 9, 10, 11, 12, 13, 14, 15 and 16 bits, respectively.

Hardware architecture of the different BinDCT stages

Each 1D-BinDCT block takes as an input eight line values, which have to be introduced simultaneously. In each of eight clock cycles, these 8 values are recovered from the serial input (Xin) and transmitted in parallel to the BinDCT stage 1 through the "input block". Fig. 2 gives the hardware structure of the input block. It consists of 8 registers used to store the 8 line values (X0–X7) to be introduced to the 1D-BinDCT block and 8 shift registers for serial to parallel transformation of the input values.

As shown in Fig. 2, this module is controlled by a counter (cntr8). Every time the counter output reaches a value of 8 (eight cycles), the load control input of the memory registers (X0-reg, …, X7-reg) is activated and the 8 recovered values are loaded into these registers. This operation takes 8 cycles for all the 8×8 input matrix lines (1–7), and only 9 cycles for the first line (line 0). In fact, the counter starts counting from zero for the first line and from one for the other lines. The load input of the counter will be activated when the counter reaches the value of 8. In this case, the binary input "0001" is loaded to be the initial counting value. Two input blocks and two counters are used in the BinDCT architectures, one for each 1D-BinDCT block. The first counter is enabled in the start of the system functioning, and the second one is enabled when the outputs of the first 1D-BinDCT block and the transpose memory are generated. The input block is the same for the different BinDCT implementation solutions.

Fig. 3a presents three different implementation solutions of stage 1: a parallel solution, a solution with 2 shared operators (1 adder and 1 subtractor), and a solution with 1 shared operator (1 add/sub). The first implementation solution uses 4 adders, 4 subtractors and 8 output registers. It has a latency of 1 cycle, as all the ai coefficients can be calculated in a parallel way. In the second implementation solution, one adder and one subtractor are used with 2 Multiplexers (MUX) to select the operators' entries and the output registers. Depending on the value of the MUX selection input "sel", two ai coefficients can be calculated every time. The calculated coefficients depend on the same Xi input. A latency of 4 cycles is necessary for calculating the ai coefficients simultaneously two by two.

The last implementation solution of stage 1 consists in using only one add/sub operator, 2 MUX for the operator entry selection, and the output registers. A signal ("alu1") is used for the add/sub component control. If alu1 = '0' then add/sub will operate as an adder; elsewhere, it will operate as a subtractor. By applying this implementation solution, 8 cycles are necessary to calculate the 8 ai coefficients. Only one coefficient can be calculated in each cycle.

The calculation order of the ai coefficients depends on the subsequent stages implementation solutions for a given architecture of the global BinDCT. The ai coefficients have to be calculated in a way that the following stages coefficients can be calculated as soon as possible without cycle loss.

For the three implemented solutions, a set of 8 registers is utilized to store the generated a_i coefficients, and each register is controlled by an "enable" (EN) signal. This permits to validate the register or registers to be loaded in each operating cycle depending on the calculation order. Therefore, the registers and consequently the corresponding coefficients can be loaded in different possible cycles, and the required number of cycles to calculate the entire coefficients can change. Accordingly, the mentioned number of cycles represents the minimum one permitted by the considered implementation solution.

For example, an adequate calculation order of the ai coefficients, using the second implementation solution, is the following one: a1 and a6, a2 and a5, a0 and a7, and a3 and a4. The given pairs of coefficients represent the only possible combinations, which can be calculated simultaneously, as they depend on the same x_i input. However, each coefficient can be calculated alone if necessary, and in this case the total number of cycles will increase. For the third solution, the following order is considered: a6, a5, a0, a3, a1, a2, a7, a4.

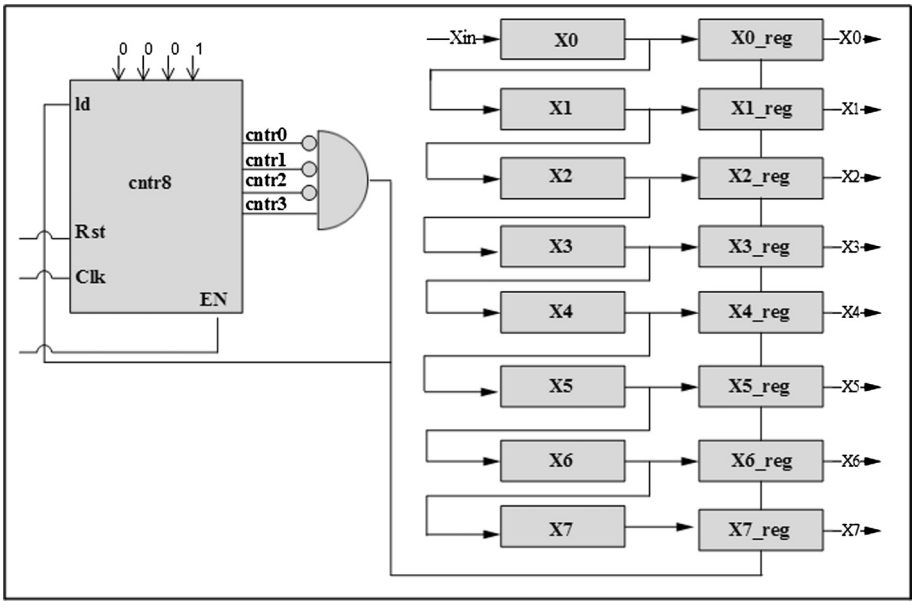


Fig. 2 Architecture of the Input block.

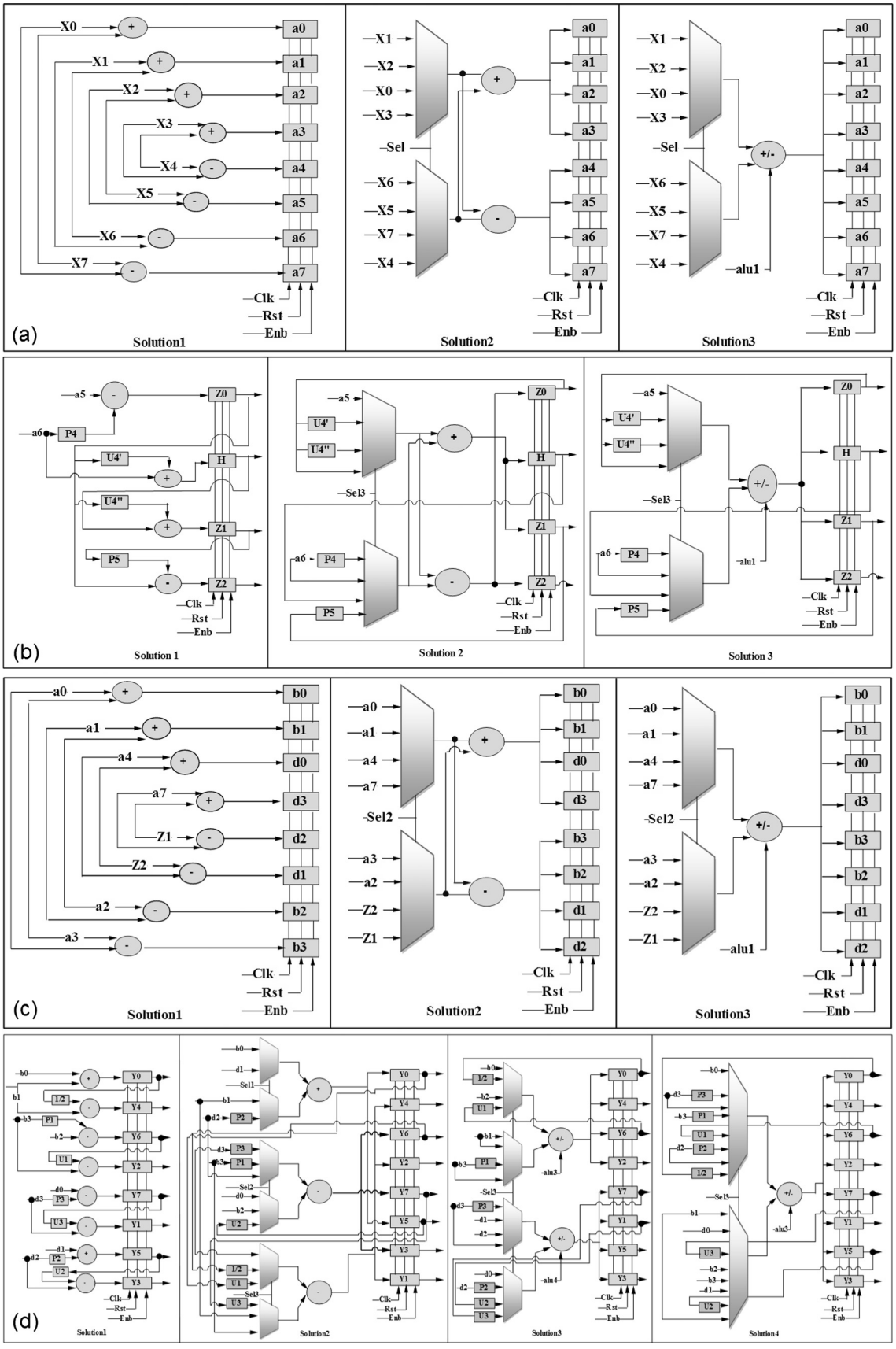


Fig. 3 Different implementation solutions of (a) stage 1, (b) stage 2, (c) stage 3, and (d) stage 4.

As depicted in Fig. 3b, the first implementation solution corresponding to stage 2 includes two adders, two subtractors, and four output registers. The second implementation solution holds 2 multiplexers, 1 adder, 1 subtractor, and the 4 output registers. In the third solution, only one add/sub operator, two MUX and 4 output registers are used. A control signal (“alu1”) is employed to select the operation (addition or subtraction). Depending on the value of the MUX selection signal, one of the Z0, H, Z1 and Z2 coefficients is calculated and the corresponding register is enabled throughout the EN input. The calculation order and the necessary number of cycles for this stage are always the same, independently from the hardware implementation solution. Four cycles are necessary to generate the two outputs of this stage: Z1 and Z2. Z1 is generated in the third cycle, and Z2 is generated in the fourth one.

Three implementation solutions are also proposed for stage 3, as shown in Fig. 3c: a parallel solution, a solution with 2 shared operators (1 adder and 1 subtractor), and a solution with 1 shared operator (1 adder/sub). These solutions have an identical structure and latency as those of the first stage. Indeed, as for stage 1, 8 outputs (b0, b1, b2, b3, b4, d0, d1, d2, d3, d4) have to be calculated, and each two output coefficients depend on the same input ones.

The minimal number of cycles for each implementation solution is 1, 4 and 8 cycles for the 1st, 2nd and 3rd solutions, respectively. For the 2nd implementation solution, the pairs of coefficients that can be calculated simultaneously are b1 and b2, b0 and b3, d0 and d1, and d3 and d2. These coefficients can be also calculated one by one if necessary. Simply, the corresponding register of each coefficient has to be enabled. In solution 3, the different coefficients can be calculated one by one.

The coefficient calculation order for the different implementation solutions is determined according to the whole BinDCT architecture and the data dependency, as it is explained in what follows.

Four hardware solutions are implemented for stage 4, as presented in Fig. 3d. 2 adders, 6 subtractors and 8 output registers are used for the first implementation solution. This latter has a minimal latency of 2 cycles since Y4, Y2, Y3 and Y1 depend respectively on Y0, Y6, Y5 and Y7. The second solution is composed of two 2-to-1 multiplexers, four 3-to-1 multiplexers, 1 adder, 2 subtractors, and the output registers. The minimal number of cycles for this solution is 3. One addition and two subtractions are possible in each cycle. The following calculation orders are the possible ones for the latency of 3 cycles:

- Y0 & Y7 & Y6 → Y5 & Y1 & Y2 → Y3 & Y4
- Y0 & Y7 & Y6 → Y5 & Y1 & Y4 → Y3 & Y2.

A solution with 2 adders and 2 subtractors can be considered; nevertheless, this solution necessitates more hardware resources than solution 2 for the same minimal number of cycles. That is why, this solution is not adopted.

4 multiplexers and 2 add/sub operators are utilized in the third implementation solution. “alu3” and “alu4” signals are employed for the add/sub operator control. The Yi outputs can be calculated in 4 cycles for different possible orders, such as the following ones:

- Y0 & Y7 → Y1 & Y6 → Y2 & Y5 → Y3 & Y4
- Y0 & Y7 → Y5 & Y6 → Y1 & Y2 → Y3 & Y4
- Y0 & Y7 → Y5 & Y6 → Y3 & Y4 → Y1 & Y2
- Y0 & Y6 → Y2 & Y4 → Y5 & Y7 → Y3 & Y1
- ...

In the fourth solution, only one add/sub operator is used with two 8-to-1 MUX and the output registers. The “alu3” input signal is opted for controlling the add/sub operator, and an addition or a subtraction is made depending on the calculated coefficient. 8 cycles are necessary to generate all the Yi coefficients.

Different calculation orders can be considered for the various implementation solutions. On the other hand, the calculation order must be fixed while respecting, as much as possible, the following priority conditions:

- Priority 1: (Y0 and Y7) → Y7 must be generated in priority to be able to calculate Y1 (using Y7) in the following cycle.
- Priority 2: (Y1 and Y6) → Y6 must be calculated in the same time or before Y1 because it will be used to calculate Y2.
- Priority 3: (Y2 & y5) → y5 must be calculated with or before Y1 because it will be utilized to calculate Y3.
- Priority 4: (Y3).
- Priority 5: (Y4) → Y4 depends on Y5.

These priority conditions cannot be always totally respected, owing to the dependency on the other stage implementation solutions. This leads to supplementary cycles necessary to generate the Yi coefficients. In general, for a given BinDCT architectural solution, the ideal coefficient calculation order is the one permitting to consecutively generate the “Yi” outputs (Y0, Y1, Y2, Y3, Y4, Y5, Y6 and Y7) in a minimal number of cycles. The coefficient calculation order for the different BinDCT stages is determined by taking into account the data dependency and the possibilities offered by the hardware

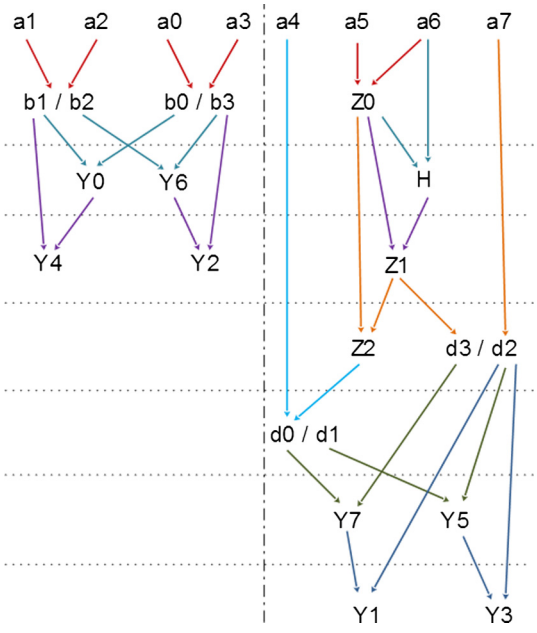


Fig. 4 Dependency graph.

Table 2 Hardware resources occupation and latency of the first and the second 1D-BinDCT stages.

	Slice Luts	Slice registers	Latency (cycles)	Slice Luts	Slice registers	Latency (cycles)	Slice Luts	Slice registers	Latency (cycles)	Slice Luts	Slice registers	Latency (cycles)
	Stage 1			*Stage 2*			*Stage 3*			*Stage 4*		
Solution 1 (S.1)	72	72	1	40	40	4	84	84	1	92	96	2
Solution 2 (S.2)	41	72	4	40	40	4	44	84	4	84	96	6
Solution 3 (S.3)	31	72	8	29	40	4	34	84	8	70	96	4
Solution 4 (S.4)										76	96	8
	Stage 11			*Stage 22*			*Stage 33*			*Stage 44*		
Solution 1 (S.1)	104	104	1	56	56	4	116	116	1	124	128	2
Solution 2 (S.2)	52	104	4	56	56	4	68	116	4	108	128	6
Solution 3 (S.3)	47	104	8	41	56	4	50	116	8	95	128	4
Solution 4 (S.4)										95	128	8

implementation solutions corresponding to the considered BinDCT architecture.

The data dependency can be considered by exploring the dependency graph represented in Fig. 4. As shown in this figure, the coefficients a5 and a6 have to be calculated in priority, since any delay on the calculation of Z0 and then on the calculation of H will introduce additional cycles on the calculation of the Y7 and Y1 outputs. Y5 presents a mobility of two cycles; i.e., it can be calculated two cycles after d1 and d2. As y3 depends on y5 and d2, it has the same mobility as y5. d1 and d2 depend respectively on a4 and z2 and z1 and a7. a4 and a7 can be then calculated 4 cycles after a5 and a6, as they can only be used from the fifth cycle after Z0. The priority order for the left side of the dependency graph (corresponding to a0, a1 a2 a3 entries) can be determined utilizing the same principle.

The possible calculation order, enabled by the different implementation solutions of the BinDCT stages, depends on their hardware architectures. This constraint should be also taken into consideration in addition to the data dependency to lead to the most optimal solutions in terms of number of cycles.

The proposed architectures of the different BinDCT stages are described in VHDL, and the Xilinx ISE tool is used to synthesize and implement the circuits on a Virtex-6 FPGA. Table 2 represents the synthesis results of the first and second 1D-BinDCT stage implementation solutions. The occupied hardware resources and the number of cycles relative to each implementation solution are given.

The operating frequency of the different stages can exceed 400 M*Hz*. Yet, the frequency of the whole 2D-BinDCT architecture is limited by the transpose memory frequency, which is equal to 378.215 M*Hz*. The architectural exploration, in the next sub-section is carried out considering that all the 2D-BinDCT architectures present the same clock frequency.

Architectural exploration of the 2D-BinDCT

The main focus of this section is the architectural exploration of the different possible implementation solutions of the 2D-BinDCT. The 2D-BinDCT architectures are obtained by combining the different implementation solutions of the BinDCT stages. The compromise between latency and hardware resource occupation is the main criterion to be taken into account when making stage combinations. If an architectural solution requires more hardware resources than another one with the same or lower number of cycles, it will be eliminated as it does not have any interest, compared to the retained architectures (either in terms of resource occupation or latency).

The architectural exploration starts with the extreme solutions in terms of hardware resources. They correspond to completely unshared and totally shared resources. For the completely unshared architecture, the first implementation solution (‘‘not shared’’) is used for each BinDCT stage. However, in the second architecture (‘‘highly shared’’), the last solution (shared with only one Add/Sub operator) is applied for each stage. Table 3 is utilized for the exploration phase. It represents the stage implementation solutions corresponding to each architecture, the latency, and the hardware resources

Table 3 Architectural solutions exploration.

	Arch N°	Stage 1/11	Stage 2/22	Stage 3/33	Stage 4/44	Number of cycles	Hardware resources (Luts)
Completely non shared solution	1	S.1	S.1	S.1	S.1	160	688
Derived solutions	2	S.1	S.3	S.1	S.1	160	662
	3	S.1	S.3	S.1	S.2	160	638
	4	S.1	S.3	S.1	S.3	160	611
	5	S.1	S.3	S.2	S.1	160	574
	6	S.1	S.3	S.2	S.2	160	550
	7	**S.1**	**S.3**	**S.2**	**S.3**	**160**	**523**
	8	S.1	S.3	S.3	S.1	162	546
	9	S.1	S.3	S.3	S.2	162	522
	10	**S.1**	**S.3**	**S.3**	**S.3**	**162**	**495**
Highly shared solution	11	S.3	S.3	S.3	S.4	170	397
Derived solutions	**12**	**S.3**	**S.3**	**S.3**	**S.3**	**170**	**391**
	13	S.3	S.3	S.3	S.2	170	405
	14	**S.3**	**S.3**	**S.3**	**S.1**	**168**	**429**
	15	S.3	S.3	S.1	S.3	166	507
	16	S.3	S.3	S.1	S.2	166	534
	17	S.3	S.3	S.1	S.1	166	558
	18	**S.3**	**S.3**	**S.2**	**S.3**	**166**	**419**
	19	S.3	S.3	S.2	S.2	166	446
	20	S.3	S.3	S.2	S.1	166	470
Medium shared solution	21	S.2	S.2	S.2	S.3	162	466
Derived solutions	**22**	**S.2**	**S.3**	**S.2**	**S.3**	**162**	**440**
	23	S.2	S.3	S.2	S.1	162	491
	24	S.2	S.3	S.2	S.2	162	467
	25	S.2	S.3	S.1	S.1	162	579
	26	S.2	S.3	S.1	S.2	162	555
	27	S.2	S.3	S.1	S.3	162	528
	28	**S.2**	**S.3**	**S.3**	**S.1**	**164**	**463**
	29	S.2	S.3	S.3	S.2	166	439
	30	**S.2**	**S.3**	**S.3**	**S.3**	**166**	**412**

Bold font is used to highlights the retained solutions.

occupied by the first and second 1D BinDCT. Only the occupation in terms of LUTs is given, as the number of registers is the same for the different architectures.

The timing diagram of the completely unshared architecture is represented in Fig. 5a. This architecture includes 4 adders, 4 subtractors and 8 registers for stage 1; 2 adders, 2 subtractors and 4 registers for stage 2; 4 adders, 4 subtractors and 8 registers for stage 3; and 2 adders, 6 subtractors and 8 registers for stage 4. The timing diagram shows the number of cycles needed for the various 1D-BinDCT stage modules. The serial inputs of the 1D-BinDCT block (X0, X1, …, X7), their serial outputs (Y0, Y1, …, Y7) and the output coefficients of each stage are provided. The first line (X0, X1, …, X7) of the 8×8 input matrix requires a latency of 16 cycles, and each one of the remaining lines presents a latency of 15 cycles. In fact, the input stage takes 9 cycles for the first line and 8 cycles for the other lines, and the four BinDCT stages have a global latency of 8 cycles.

The first valid output of the 2D-Bindct, relative to the first architecture, is generated after 97 cycles, and the last one is generated after 160 cycles. These values are calculated for the serial inputs and outputs of the 2D-BinDCT, and they are obtained considering the pipeline between the different BinDCT stages, the transpose memory and the additional blocks for output serialization.

Relative to the first architecture and its timing diagram, the different derived 2D-BinDCT implementation solutions (Table 3) are explored. The following considerations are taken into account in the exploration phase:

- For stages 2/22, each one of the proposed solutions (S.1, S.2, S.3 and S.4) takes 4 execution cycles. That is why only the solution S.3 is opted for, because it has the best performances in terms of resource occupation.
- The implementation solution S.4 of stages 4/44 is not considered, because it necessitates more execution cycles and more hardware resources than the solution S.3. Thus, for the shared solutions in stage 4, only S.2 and S.3 are employed.

The first architectural solution, combining the S.1, S.1, S.1 and S.1 solutions for stages 1, 2, 3 and 4, is automatically replaced by the S.1, S.3, S.1 and S.1 solution-based architecture. For this latter (Arch N°2), the different combinations of the S.1, S.2 and S.3 solutions of stages 3/33 and the S.1, S.2 and S.3 solutions of stages 4/44 are explored. The retained architectural solutions are Arch N°7 and Arch N°10, which are indicated in Table 3. The remaining architectures are discarded as there exists solutions with the same number of cycles and fewer hardware resources.

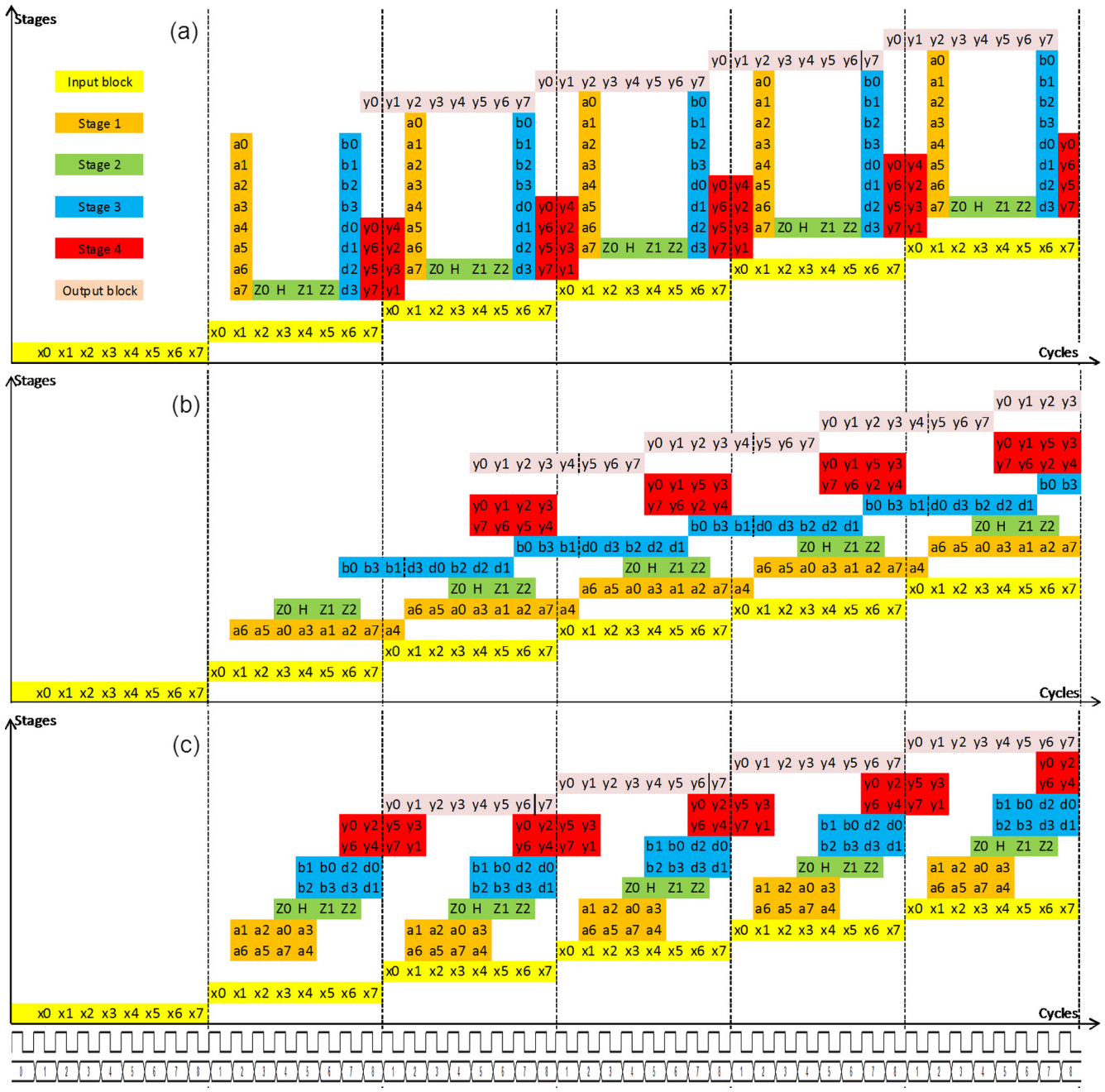


Fig. 5 Timing diagram of (a) Arch N°1, (b) Arch N°12 and (c) Arch N°21.

For the highly shared architecture (Arch N°11), only 1 add/sub operator is used for each BinDCT stage. According to the previously mentioned considerations, this architecture, which is built utilizing respectively the S.3, S.3, S.3 and S.4 solutions of stages 1, 2, 3 and 4, is automatically replaced by the S.3, S.3, S.3 and S.3 solution-based architecture (Arch N°12). The timing diagram corresponding to Arch N°12 is represented in Fig. 5b. The first valid output of the 2D-BinDCT, relative to this architecture, is generated after 107 cycles, and the last one is generated after 170 cycles.

The different implementation solutions derived from this architecture (Arch N°12) are explored. Only the S.3 implementation solution is used for stage 2, and the S.1, S.2 and S.3 solutions are utilized for stage 4. The retained architectural solutions are Arch N°12 and Arch N°18.

The architecture based on the S.2, S.2, S.2 and S.3 implemention solutions for stages 1, 2, 3 and 4 is also considered as an architectural exploration starting point. In Table 3, this architecture is called "medium shared solution". Each one of stages 1 to 3 includes 1 adder and 1 subtractor, and just stage 4 includes 2 add/sub operators. The timing diagram corresponding to this solution (Arch N°21) is illustrated in Fig. 5c. The first valid output of the 2D-BinDCT, relative to this architecture, is generated after 99 cycles, and the last

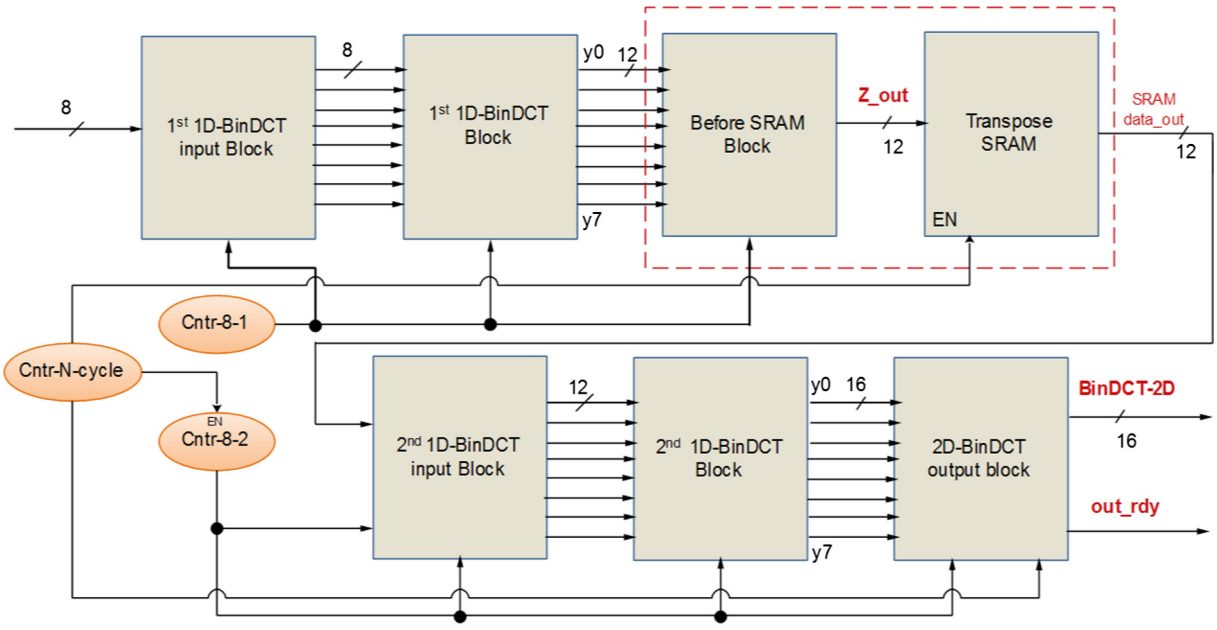


Fig. 6 2D-BinDCT architecture.

Table 4 Implementation results of the retained architectures.

Architecture	Slice registers	Slice LUTs	Slices	Frequency M*Hz*	Number of cycles
Arch N°7	1801	938	665	378.215	160
Arch N°10	1809	925	638	378.215	162
Arch N°22	1801	846	626	378.215	162
Arch N°28	1809	890	645	378.215	164
Arch N°18	1809	834	661	378.215	166
Arch N°30	1809	833	613	378.215	166
Arch N°14	1817	870	595	378.215	168
Arch N°12	1817	813	588	378.215	170

one is generated after 162 cycles. By exploring the different architectural solutions derived from Arch N°21, the retained ones are Arch N°22, Arch N°28 and Arch N°30.

Results and discussion

The implementation details of the 2D-BinDCT and the discussion of results are presented in this section, to wit the structure of the global architecture, the operations of the proposed architecture, the implementation results in terms of hardware resources and clock frequency, and the evaluation and comparison of performances. The suggested architecture is depicted in Fig. 6. It includes the following blocks: the "1st 1D-BinDCT input" block, the "1st 1 D-BinDCT" block, the "before SRAM" block, the "transpose SRAM" block, the "2nd 1D-BinDCT input" block, the "2nd 1D-BinDCT" block and the "2D-BinDCT output" block.

The 1st and 2nd 1D-BinDCT input blocks have the same structure, which is indicated in Fig. 2. Their role is to transmit the 8 serially introduced inputs in a parallel way to the first BinDCT stage. These blocks operate in pipeline with the other architecture blocks and present an initiation period of 1 cycle. Accordingly, data acquisition is ensured in each clock cycle, which is very important for the continuous data acquisition for a direct connection to a video source or for a continuous access to the data memory.

The "before SRAM" block permits serializing the parallel outputs of the first 1D-BinDCT and transmitting them to the transpose memory block. The 1D-BinDCT coefficients have to be transmitted in a continuous way in the ascending order (Y0, …, Y7). Similarly, the "2D-BinDCT output" block allows serializing the parallel outputs of the 2nd 1D-BinDCT and putting them out. This is essential for pixel-by-pixel transmission and acquisition.

Table 4 gives the implementation results of the complete architecture corresponding to the BinDCT architectural solutions retained in the exploration phase. As shown in this table, for the same operating frequency, the Arch N°10 and Arch N° 22 have the same number of cycles but a different hardware resource occupation. Since Arch N°22 presents a smaller size, Arch N°10 is discarded. Likewise, Arch N°18 is also eliminated, as it provides a higher size than Arch N°30 for the same number of cycles. Comparing Arch N°22 and Arch N°28, it can be noticed that Arch N°22 has a smaller size and a lower number of cycles, so Arch N°28 is discarded. In the same way,

Table 5 Control signals for the different BinDCT stages of architecture N°12.

Mux1 output	Mux2 output	Output	Operation	Alu	Sel	Enb	Cntr8
a. Stage 1 control							
x1	x6	a6	x1 − x6	1	00	000	(0001)
x2	x5	a5	x2 − x5	1	01	001	(0010)
x0	x7	a0	x0 + x7	0	10	010	(0011)
x3	x4	a3	x3 + x4	0	11	011	(0100)
x1	x6	a1	x1 + x6	0	00	100	(0101)
x2	x5	a2	x2 + x5	0	01	101	(0110)
x0	x7	a7	x0 − x7	1	10	110	(0111)
x3	x4	a4	x3 − x4	1	11	111	(1000)
b. Stage 2 control							
a5	a6.P4	Z0	a5 − a6.P4	1	00	00	(0011)
a6	Z0.U4′	H	a6 + Z0.U4′	0	01	01	(0100)
H	Z0.U4″	Z1	H + Z0.U4″	0	10	10	(0101)
Z1.P5	Z0	Z2	Z1.P5 − Z0	1	11	11	(0110)
c. Stage 3 control							
a0	a3	b0	a0 + a3	0	00	000	(0110)
a0	a3	b3	a0 − a3	1	00	001	(0111)
a1	a2	b1	a1 + a2	0	01	010	(1000)
a4	Z2	d0	a4 + Z2	0	10	011	(0001)
a7	Z1	d3	a7 + Z1	0	11	100	(0010)
a1	a2	b2	a1 − a2	1	01	101	(0011)
a7	Z1	d2	a7 − Z1	1	11	110	(0100)
a4	Z2	d1	a4 − Z2	1	10	111	(0101)

Mux1 output	Mux2 output	Mux3 output	Mux4 output	Output	Operation	Sel	Enb	Alu1	Alu2	Cntr8
d. Stage 4 control										
b0	b1	P3.d3	d0	Y0	b0 + b1	00	00	0	1	(0100)
				Y7	P3.d3 − d0					
d3	Y7.U3	b3.P1	b2	Y1	d3 − Y7.U3	01	01	1	1	(0101)
				Y6	b3.P1 − b2					
Y6.U1	b3	P2.d2	d1	Y2	Y6.U1 − b3	10	10	1	0	(0110)
				Y5	P2.d2 + d1					
d2	Y5.U2	½Y0	b1	Y3	d2 − Y5.U2	11	11	1	1	(0111)
				Y4	½Y0 − b1					

Table 6 Hardware performances comparison to different DCT implementations.

L	LUTs	Registers	F max
Llamocca [23]			
12	5167	4866	250
16	8609	7269	244
Bouguezel et al. [24]			
12	821	1523	337.8
16	1029	1915	284.0
Bouguezel et al. [11] for α = 0			
12	728	1732	337.4
16	919	2187	325.2
Bouguezel et al. [11] for α = 1			
12	813	1949	329.4
16	1021	2445	316.9
Bouguezel et al. [11] for α = 2			
12	812	1950	353.1
16	1019	2445	326.5
Cintra and Bayer [12]			
12	950	1709	270.6
16	1198	2162	256.0
Bayer and Cintra [13]			
12	657	1329	354.0
16	834	1698	345.5
Potluri et al. [14]			
12	1036	1968	314.9
16	1291	2463	298.0
Potluri et al. [7]			
12	663	1329	353.7
16	839	1697	341.8
Proposed BinDCT implementation (Arch 22)			
16	815	1608	378.215

Arch N°14 presents a higher number of cycles and a larger size compared to Arch N°30, so it is also discarded.

As a final result, Arch N°12 is retained as the best solution in terms of hardware resource occupation, Arch N°7 as the best solution in terms of number of cycles, and Arch N°22 and Arch N°30 as two solutions both assuring the best compromise between the number of cycles and the hardware resource occupation. Depending on the target application constraints, the user has the choice between one of these implementation solutions, namely the solution optimizing hardware resources (Arch N°07), the solution optimizing latency (Arch N°12), or one of the intermediate solutions (Arch N°22 and Arch N°30).

An additional advantage of the proposed implementation architectures of the BinDCT is their simple control mechanism. In fact, the whole system control is just ensured by the "cntr-8-1", "cntr-8-2" and "Cntr-N-Cycle" counters, so no additional controllers (state machines) are needed. The counter "Cntr-N-Cycle" is used to enable the transpose memory and the cntr-8-2. The transpose memory is enabled when the first 1D-BinDCT coefficient is generated and sent throughout the "before SRAM" block. The "Cntr-N-Cycle" is also used to generate the "out_rdy" signal. As it can be seen in the timing diagrams, the different intermediate coefficients (ai, zi, bi, di and yi) are generated in a periodic way relative to the cntr-8-1 values. This latter is utilized to control the different multiplexers and registers of the first 1D_BinDCT architecture for data selection and register load activation. Cntr8-2 is used for the second 1D_BinDCT control, in the same way as cntr8-1. Nevertheless, this counter does not begin working with the system start. It is enabled when all the 1D-BinDCT coefficients are calculated and transposed. In this time, the 2nd 1D-BinDCT input block must begin getting the input values.

The control signals (Alu, Sel, En, Alu1, etc.) of the different BinDCT stage modules are directly commanded by the "cntr-8" counter. In each counting cycle, some operations are to be performed to calculate particular coefficients, as defined in the timing diagram related to each implementation solution. Table 5 gives the control words and the performed operations for stages 1, 2, 3 and 4. These control steps correspond to the functioning example of Arch N°12, which is represented by the timing diagram of Fig. 5b. In each control step, only 2 or 3 less significant bits of the cntr-8 are used to command the multiplexers and registers to assure the defined operation, as indicated in Table 5.

The obtained results in terms of number of cycles correspond to the execution of one 8×8 input matrix. This may be the case, for example, of the MPEG7 color layout descriptor [21,22], when the DCT is calculated for only one 8×8 matrix for each color component (Y, Cr and Cb). Whereas, for a continuous pipelined execution of the BinDCT for the different 8×8 pixels blocks of the input images, just a small change will be introduced to the architecture. A second transpose memory block with a demultiplexer after the "Before-SRAM" block and a multiplexer just after the two "transpose memory" blocks is to be added. These two transpose memories are going to be used to alternately store the intermediate results generated by the first 1D-BinDCT module. When the second BinDCT module is reads data from one memory block, the first 1D-BinDCT module will write into the other one. After the required number of cycles to make the first BinDCT coefficient available on the out port, the outputs will be generated in every clock cycle.

For a continuous execution mode, our architectures can attain a throughput of 378 Mega pixels per second (Mpps). This throughput can increase if a more recent FPGA family has been used (for example, it becomes 390 Mpps for a Virtex 7 FPGA). This high throughput permits supporting a resolution from the VGA (640×480 pixels) to the UXGA (1600×1200) and all digital television formats including the HDTV (1080i and 720p). For example, the VGA format with 60 frames per second (fps) requires a throughput of only 30.6 Mpps. The 1080p/60 video format (1920×1080 pixels @ 60 *Hz* refresh) needs a bandwidth of 148.5 Mpps, which is still remarkably less than the throughput supported by our architectures.

The performance of the proposed implementation of the BinDCT is compared to the existing DCT implementations in terms of hardware cost and computing time (Table 6). The hardware cost is measured by the number of look-up tables (LUT) and registers (Regs) and the computing time is normalized as clock cycles (maximum operating frequency (Fmax) in M*Hz*). The DCT architectures used for comparison are those suggested by Potluri et al. [7], Dhandapani and Ramachandran [8], and Llamocca et al. [23]. The architectures

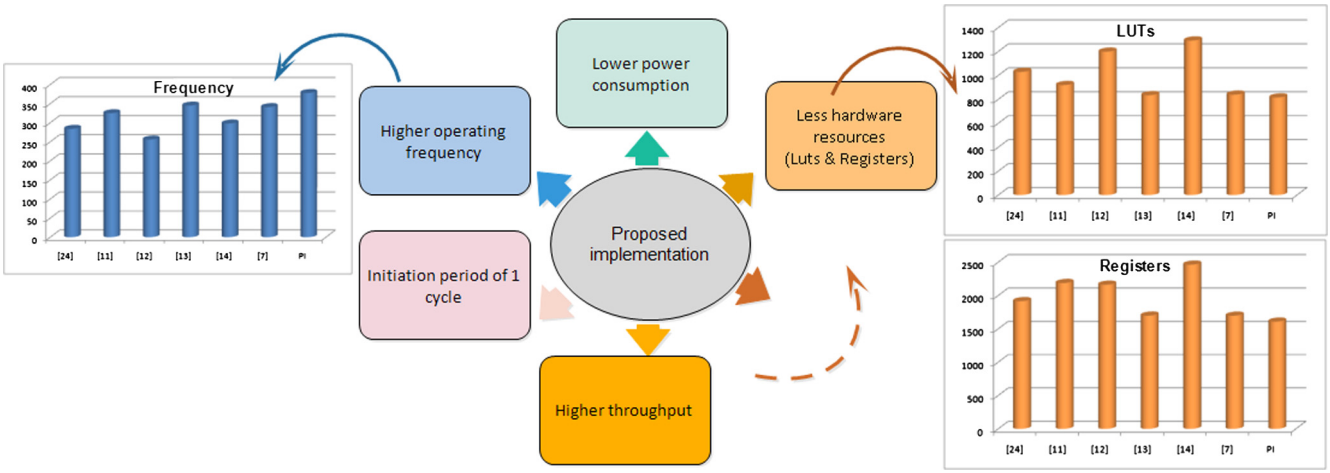


Fig. 7 Proposed implementation (PI) performances.

implemented by Potluri et al. and Dhandapani and Ramachandran [7,8] corresponded to the 8-point multiplication free approximate DCT methods put forward by Potluri et al. [7,14], Bouguezel et al. [11,24], Cintra et al. [12], and Bayer and Cintra [13]. The architecture implemented by Llamocca et al. [23] corresponded to a distributed arithmetic DCT based implementation.

Two input precision levels have been considered for the different architectures in order to investigate the performance in terms of digital logic resource consumptions at varied degrees of numerical accuracy and dynamic ranges. The two following system word lengths have been adopted: $L \in \{12, 16\}$. The implemented architectures have been designed for 8 parallel inputs and outputs. Therefore, the same configuration is adapted for our BinDCT architecture (Arch N°22) by discarding the input and output blocks. The implementation is performed on the Xilinx Virtex-6 XC6VSX475T device, which contains 297,600 LUT and 595,200 Regs. By analyzing the obtained results, it can be seen that the BinDCT design presents the best performances in terms of hardware resources and operating frequency.

In addition, the static (Qp) and dynamic power (Dp) consumptions are estimated using the Xilinx XPower Analyzer. The following results are obtained: Qp = 3.532 mW and Dp = 0.107 mW. These consumption values represent lower values than those presented in the literature [7,8,23]. Fig. 7 summarizes the performances of the proposed implementation compared to the existing implementations of DCT approximations. The obtained performances show that the proposed implementation can be a good candidate for highly constrained images and video processing applications. It allows meeting the real time constraints of the most recent high resolution video formats, while presenting high hardware efficiency.

Conclusions

In order to develop an efficient VLSI architecture for the BinDCT, in this work, a large design exploration of this module is performed. At first, a detailed study of the BinDCT, which is decomposed in a multi-stage architecture, is carried out. Several architectures of the whole 2D-BinDCT are developed by exploring different BinDCT stage hardware implementation solutions. These architectures are obtained by combining the different implementation solutions of the BinDCT stages. The timing of the explored solutions is determined by taking into account the stages pipeline and the coefficients calculation order. This latter is fixed in the manner of ensuring the best latency while avoiding data dependency violation.

The hardware architectures, relative to the different retained solutions, are designed and implemented. The control system is also studied and developed. The implementation results for a Virtex-6 FPGA show the high performances of the proposed system compared to the existing DCT designs, which makes it suitable for hardware and time constrained applications.

Conflict of Interest

The authors have declared no conflict of interest.

Compliance with Ethics Requirements

This article does not contain any studies with human or animal subjects.

References

[1] Turneo A, Monchiero M, Palermo G, Ferrandi F, Sciuto D. A pipelined fast 2D-DCT accelerator for FPGA-based SoCs. Proc - IEEE Comput Soc Annu Symp VLSI Emerg VLSI Technol Archit 2007:331–6.
[2] Road M, Kumar PP, Road M. FPGA implementation of a DA based 1D DCT processor. Int J Rev Electron Commun Eng 2013;1:94–6.
[3] Venkata B, Venkateswarlu C. Design of low power 2-D DCT architecture using reconfigurable architecture. IOSR J Electron Commun Eng 2012;3:20–5.
[4] Liang J, Tran TD. Fast multiplierless approximations of the DCT with the lifting scheme. IEEE Trans Signal Process 2001;49:3032–44.
[5] Cham WK. Development of integer cosine transforms by the principle of dyadic symmetry. IEE Proc I - Commun Speech Vis 1989;136:276–82.

[6] Tran TD. The BinDCT: fast multiplierless approximation of the DCT. IEEE Signal Process Lett 2000;7:141–4.
[7] Potluri US, Madanayake A, Cintra RJ, Bayer FM, Kulasekera S, Edirisuriya A. Improved 8-point approximate DCT for image and video compression requiring only 14 additions. IEEE Trans Circuits Syst I 2014(61):1727–40.
[8] Dhandapani V, Ramachandran S. Area and power efficient DCT architecture for image compression. EURASIP J Adv Signal Process 2014;2014:180. http://dx.doi.org/10.1186/1687-6180-2014-180.
[9] Meher PK, Park SY, Mohanty BK, Lim KS, Yeo C. Efficient integer DCT architectures for HEVC. IEEE Trans Circuits Syst Video Technol 2014;24:168–78.
[10] Bouguezel S, Ahmad MO, Swamy MNS. Low-complexity 8 times 8 transform for image compression. Electron Lett 2008;44:1249–50.
[11] Bouguezel S, Ahmad MO, Swamy MNS. A low-complexity parametric transform for image compression. In: 2011 IEEE Int Symp Circuits Syst. p. 2145–8.
[12] Cintra RJ, Bayer FM. A DCT approximation for image compression. IEEE Signal Process Lett 2011;18:579–82.
[13] Bayer FM, Cintra RJ. DCT-like transform for image compression requires 14 additions only. Electron Lett 2012;48:919–21.
[14] Potluri US, Madanayake A, Cintra RJ, Bayer FM, Rajapaksha N. Multiplier-free DCT approximations for RF multi-beam digital aperture-array space imaging and directional sensing. Meas Sci Technol 2012;23:114003.
[15] Dang PP, Chau PM, Nguyen TQ, Tran TD. BinDCT and its efficient VLSI architecture for real-time embedded applications. J Imaging Sci Technol 2005;49:124–37.
[16] Jabbar MH. BinDCT design and implementation on FPGA with low power architecture. Liverpool: John Moores University; 2008.
[17] Murphy CW, Harvey DM. Reconfigurable hardware implementation of BinDCT. Electron Lett 2002;38:1012–3.
[18] Al-Gherify MFK. Image compression using BinDCT for dynamic hardware FPGA's. Liverpool: John Moores University; 2007.
[19] Timakul S, Chuntree S, Choomchuay S. A low complexity implementation of a fast BinDCT. In: Int symp commun inf technol (ISCIT 2003), vol. 2, Songkla, Thailand. p. 799–802 [n. d.].
[20] Mijic S, Mezei I, Struharik R. IP cores for 2D direct and inverse discrete cosine transformation. 2015 23rd Telecommun forum telfor (TELFOR); 2015. p. 433–6. http://dx.doi.org/10.1109/TELFOR.2015.7377500.
[21] Kasutani E, Yamada A. The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. Proceedings of int conf image process, vol. 1. p. 674–7.
[22] Manjunath BS, Ohm JR, Vasudevan VV, Yamada A. Color and texture descriptors. IEEE Trans Circuits Syst Video Technol 2001;11:703–15.
[23] Llamocca D, Pattichis M, Carranza C. A framework for self-reconfigurable DCTs based on multiobjective optimization of the power-performance-accuracy space. In: 2012 7th int work on reconfigurable commun syst (ReCoSoC); 2012. p. 1–6.
[24] Bouguezel S, Ahmad MO, Swamy MNS. Low-complexity 8×8 transform for image compression. Electron Lett 2008;44:1249.