Chapter 5

A Constant Throughput Architecture

This chapter presents a constant throughput 32-point integer DCT architecture to achieve almost equal processing time at each quadtree depth level. The proposed 1D DCT architecture uses resource sharing technique to compute single 32-point or two 16-point or four 8-point or eight 4-point DCTs at a time. Therefore, it is possible to attain a constant throughput, 32 samples per clock, irrespective of the transform size. The proposed modified 2D DCT architecture requires $2N + 1$ clock cycles to process $\frac{32}{N}$ blocks of $N \times N$ pixels.

5.1 Introduction

HEVC is the most complex encoder and RDO [94] is one of the most complicated tasks in it. This task takes many decisions in order to produce optimum coding

efficiency. TU size selection is one of them. Residual blocks are recursively partitioned into transform blocks during RDO process and they are represented by RQT as discussed in Chapter 1. Thus, the size of the TU varies from 4×4 to 32×32. All the RQT levels are tested to decide the optimum TU size during RDO process. Therefore, RD cost is computed several times for each quadtree structure: once for the 32×32 TU, 4 times for the 16×16 TU, 16 times for the 8×8 TU and 64 times 4×4. These variable block-size transforms and variable RQT depth increase computational complexity as well as processing time. To reduce this computational burden, some early termination algorithms are proposed [114, 115] which compare transformed coefficients with respect to a threshold value. However, those algorithms also need to compute RD cost at each RQT level for the worst case.

A statistic has been presented in Chapter 4 (Table 4.1) revealing the rate of the different size of DCTs computed during video encoding using HEVC reference software. From this table It is observed that for any encoder configuration lower order of DCTs are performed very often than that of the higher-order irrespective of the video resolution. On an average more than 50% of times 4-point DCT is performed, whereas 32-point DCT is performed less than 2% of time. The reason behind this is that the RD cost is calculated by splitting each block into four child blocks and each block is transformed iteratively [11]. It implies that most of the time RD cost is calculated up to the highest depth level of RQT. However, the throughput of the transform core varies according to the size of TU and it is the least for the lower order DCT at the highest RQT depth. Therefore, an efficient high throughput architecture is required especially for lower order DCT.

Lot of work has been done to realize DCT architecture in VLSI. The primary focus of those articles is to handle different size of transforms with high throughput and to reduced the hardware complexity. However, most of the those designs follow the

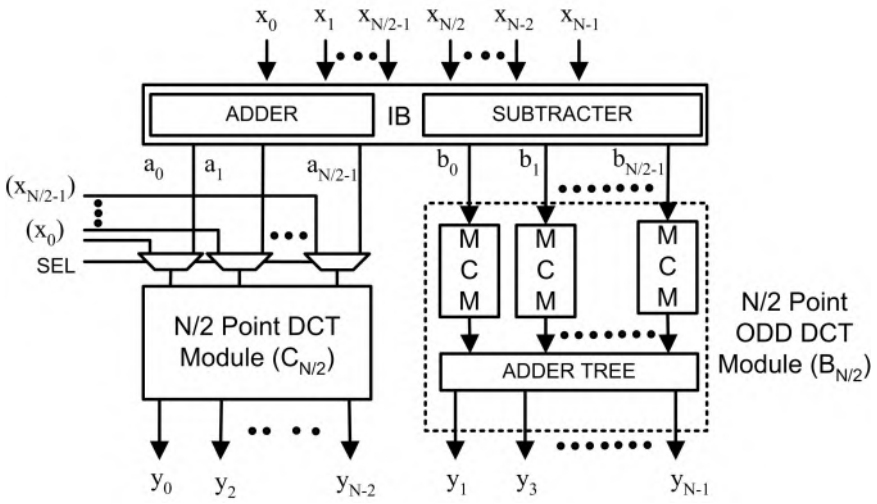

FIGURE 5.1: A generalized model for N-point 1D DCT

hierarchical structure in which $\frac{N}{2}$-point DCT module is embedded in N-point DCT architecture as shown in Fig. 5.1. Hence, a single kernel of 32-point DCT is used in which all the lower size transforms are embedded [66]. Fig. 3.6 in Chapter 3 depicts a typical hardware sharing model used in HEVC core transform [66]. It uses hardware of even coefficient calculation to compute lower order DCT. Rest of the hardware resources remain idle during this time. Hence, majority of the hardware resources of the transform core remain idle during 4 and 8 -point DCT computations and the throughput varies with TU size.

In [68], an additional $\frac{N}{2}$-point unit is used with each N-point DCT unit to achieve constant throughput irrespective of the TU size. This necessitates extra hardware. A reconfigurable architecture is proposed in [73] wherein an approximate DCT like transform is used and any N-point transform can be derived from a pair of $\frac{N}{2}$-point transform. In [74], an approximate 4-point DCT is used as a basic unit and all higher order DCTs are derived from it, such that any N-point unit can be utilized as two $\frac{N}{2}$-point units. However, approximation compromises with the DCT properties and results in low quality video.

To support high resolution for future video coding, the transform architecture throughput can be increased by efficient use of existing hardware resources. This chapter presents a DCT architecture for HEVC which is based on hardware sharing methodology to achieve high throughput with the minimum number of logic resources. It also provides almost constant throughput for all size of the TU. The advantages of the proposed architecture are as follows:

1. The proposed architecture requires almost equal processing time at each RQT level which improves the speed of the RDO process.

2. Significant energy is consumed by transform and quantization unit during RDO process as HEVC profiling results show that a large amount of time is spent in the TComTrQuant class [14]. As the proposed architecture reduces the processing time, energy consumption reduction is possible by adopting methods like clock-gating, data-gating, etc.

3. Almost all the hardware resources are fully utilized irrespective of the TU size. Consequently, the throughput of the DCT architecture remains constant and average power consumption per unit throughput remains the same for all size of the TU.

5.2 Proposed Architecture

A method to share datapath of 1D DCT architecture and its hardware implementation are discussed in this section. Fig. 5.2 shows the hardware resource sharing scheme employed in C_N unit. The method is applied on integer DCT for its popularity in HEVC. Typically, IB unit uses separate adders and subtractors in any

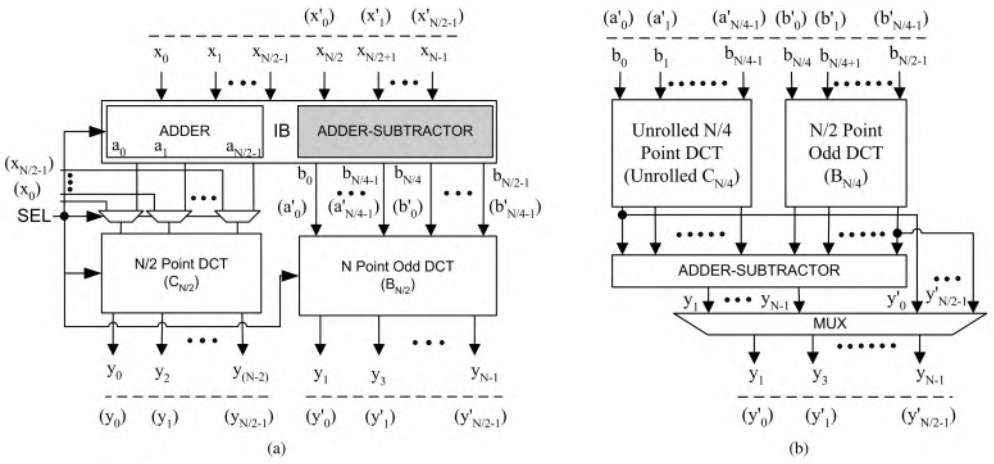

FIGURE 5.2: Shared datapath for N-point (a) DCT Module (b) Proposed Odd DCT Module

hierarchical DCT architecture [68], as shown in Fig. 5.1. However, in the proposed architecture, all the subtractors in IB unit have been replaced by adder–subtractors so that it can produce all a_n and b_n during $\frac{N}{2}$-point DCT. At this instance, subset of the input lines $\{x_{\frac{N}{2}}, x_{\frac{N}{2}+1}, \cdots, x_{N-1}\}$ behave like other input set $\{x'_0, x'_1, \cdots, x'_{\frac{N}{2}-1}\}$. To clear this point, these terms are written within bracket in Fig. 5.2(a). The datapath of $B_{\frac{N}{2}}$ is shared with an unrolled $C_{\frac{N}{4}}$ and a $B_{\frac{N}{4}}$ unit to realize an extra $C_{\frac{N}{2}}$ unit as shown in Fig. 5.2(b). Thus, hardware used to compute odd coefficients of a N-point DCT can compute a complete $\frac{N}{2}$-point DCT. Similarly, $B_{\frac{N}{4}}$ can be further shared with next lower order DCT. For example, the datapath of B_{16} is shared with an unrolled C_8 and a B_8. Further, a B_8 can be shared with an unrolled C_4 and a B_4 and so on. This resource sharing process continues up to 4-point DCT.

An unrolled $\frac{N}{4}$-point DCT can be used to produce two $\frac{N}{8}$-point DCTs by using an additional multiplexer. The equation for even coefficients in an unrolled $\frac{N}{4}$-point

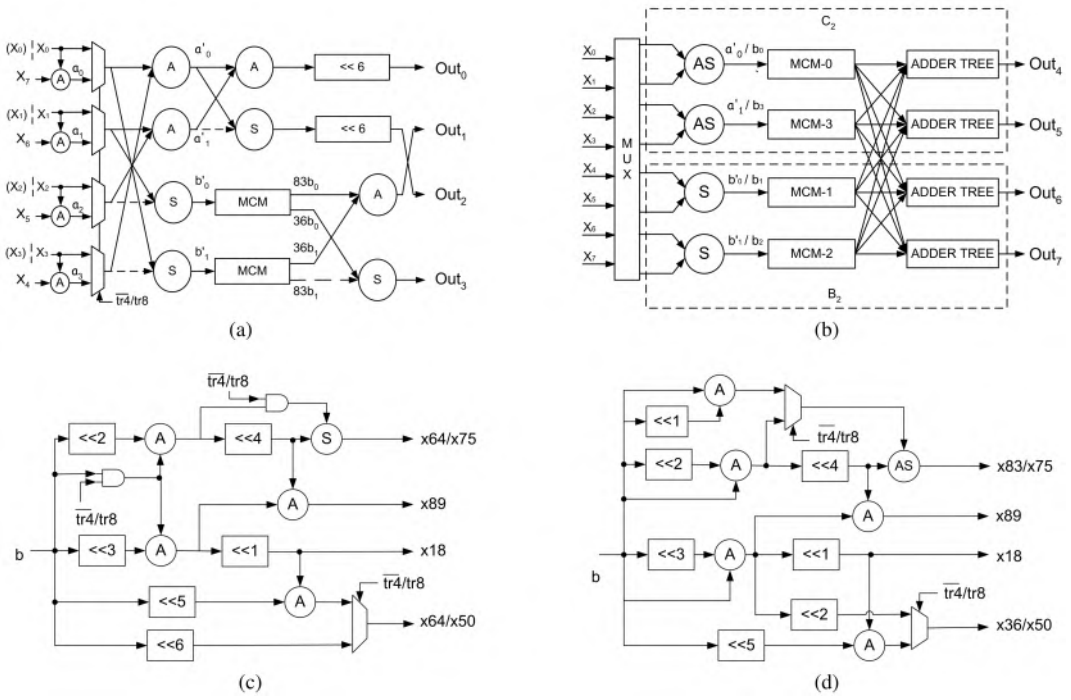

FIGURE 5.3: Shared datapath for 8-point DCT Module (a) Even unit (b) Odd unit (c) MCM-0 and MCM-3 (d) MCM-1 and MCM-2

DCT can be written as

$$[Y_{2n}^{\frac{N}{4}}] = \left[C_{\frac{N}{8}}\right]\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\frac{N}{8}-1} \end{bmatrix} + \left[C_{\frac{N}{8}}\right]\begin{bmatrix} x_{\frac{N}{4}-1} \\ x_{\frac{N}{4}-2} \\ \vdots \\ x_{\frac{N}{8}} \end{bmatrix} = E_0 + E_1, \tag{5.1}$$

where E_0 and E_1 represent two different outputs of $\frac{N}{8}$-point DCT iff $x_0, x_1, \ldots, x_{\frac{N}{8}-1}$ and $x_{\frac{N}{4}-1}, x_{\frac{N}{4}-2}, \ldots, x_{\frac{N}{8}}$ represent two different inputs, respectively. Thus, a N-point odd DCT unit can be used as either one $\frac{N}{2}$-point or two $\frac{N}{4}$-point or four $\frac{N}{8}$-point DCT units. A two bits input signal SEL is used to select the size of the DCT. An example of 8-point DCT architecture has been discussed in the following subsection.

5.2.1 8-point DCT architecture

A 8-point DCT comprises of a C_4 and a B_4 unit to compute even and odd coefficients, respectively. Usually, only C_4 unit is recursively used to calculate DCT of lower size kernel and B_4 unit remains idle [68]. It is because out of eight inputs $\{x_0, x_1, \cdots, x_7\}$ only four inputs $\{x_0, x_1, x_2, x_3\}$ are multiplexed with $\{a_0, a_1, a_2, a_3\}$, as shown in Fig. 5.3(a). Here, value of $\{a_0, a_1, a_2, a_3\}$ is calculated according to (3.25). Select signal $\overline{tr4}/tr8$ is used to choose an appropriate set of inputs depending on the size of transform under process.

If B_4 unit remains idle, its all hardware resources are underutilized. To avert this situation, B_4 unit is also used to compute another 4-point DCT in the proposed architecture. Further, its datapath is shared with an unrolled C_2 and B_2 units as shown in Fig. 5.3(b). Hardware resources in the B_4 unit are also exploited with additional multiplexers. However, the MCMs must be modified as DCT coefficients differ depending on the operation to be performed. For this purpose, ith and $(N - 1 - i)$th MCMs are included in the same datapath due to following relation.

$$|b_{ij}| = |b_{(N-i-1)(N-j-1)}| \tag{5.2}$$

Here, b_{ij} represents (i, j)th coefficient in matrix $B_{\frac{N}{2}}$ and ith MCM produces all the b_{ij} coefficients. Therefore, MCM-0 and MCM-3 are included in the datapath shared by B_4 or unrolled C_2. These MCMs are capable of performing multiplication with constant sets $\{89, 75, 50, 18\}$ and $\{64, 64\}$ during 8- and 4- point DCT, respectively. Similarly, MCM-1 and MCM-2 are included in the datapath shared by B_4 and B_2 i.e. they perform multiplication with constant sets $\{89, 75, 50, 18\}$ and $\{83, 36\}$ during 8- and 4- point DCT, respectively. Hence, different type of MCMs are required.

Fig. 5.3(c) and 5.3(d) depict modified architecture of these MCMs. A control signal $\overline{tr4}/tr8$ is used to select the type of DCT to be performed.

Here, outputs of the IB unit are fed to the MCMs and those are also shared to produce all b_n during 8-point DCT or all a_n and b_n for 4-point DCT. To accomplish this, subtractors in the IB unit are replaced by add-sub units and multiplexers are used at the input side. At the output tree, some of the adders and subtractors are replaced with the add-sub units to perform addition or subtraction operation during different size of DCT calculation.

5.2.2 Higher order DCT architecture

Aforementioned datapath sharing method has been adopted in the proposed DCT architecture and almost all resources are fully utilized. Complete datapath of 32-point odd DCT unit is shared with unrolled C_8 and B_8. Similarly, B_8 is further shared with C_4 and B_4 and so on. This datapath sharing scheme is shown in Fig. 5.4. It is clear from this figure that a 32-point odd DCT unit is divided into four datapaths. Similarly, 16-point odd DCT unit gets divided into three datapaths as represented by the shaded area in Fig. 5.4. These datapaths produce all necessary intermediate values or DCT coefficients depending on the type of DCT applied. For example, datapaths P_2, P_3, P_4 altogether produce two sets of 4-point DCT coefficients or a single set of 8-point DCT coefficients or 16-point odd DCT coefficients or intermediate values for 32-point odd DCT coefficients depending on the control signal.

In the proposed architecture datapath sharing is achieved by

1. Reconfiguring MCM units

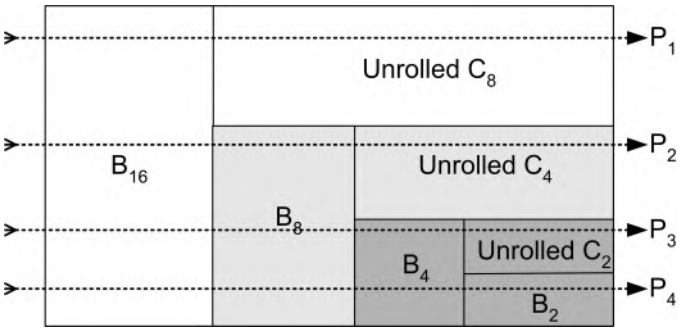

FIGURE 5.4: Proposed datapath of a 32-point odd DCT unit

2. Sharing IB unit and

3. Sharing output adder trees

5.2.2.1 Reconfigurable MCM design

There are 16 MCMs in a 32-point DCT [68], out of those 8 lie in datapath P_1 and rest of them lie in datapath P_2, P_3 and P_4. From Fig. 5.4 it is clear that the path P_1 comprises of B_{16} and unrolled C_8 units. Hence, MCMs in this path must be capable of producing two sets of coefficients. However, path P_2 comprises of B_{16}, B_8 and unrolled C_4. It means the MCMs in this path must be capable of producing three sets of coefficients. It is also clear from Fig. 5.4 that MCMs utilized in P_3 and P_4 must be capable of computing coefficients for all size of DCT. Consequently, they form the critical path.

Criteria to select MCMs for a particular datapath has been discussed in subsection 5.2.1. Datapaths in a 32-point odd DCT unit and MCMs corresponding to them are listed in Table 5.1. Here, M_i represents ith MCM responsible for generating all the b_{ij} coefficients. There is slight variation in the MCMs used in a particular datapath. Therefore, there is a separate type of MCM for each datapath.

Constant multiplication in each MCM is realized by A-operation [69] which can be written as

$$A(u, v) = |2^{l_1}u + (-1)^s 2^{l_2}v|2^{-r}. \tag{5.3}$$

Here, u and v are already realized or intermediate constants. l_1, l_2, and r are positive integers. Here, $s \in \{0, 1\}$ decides either addition or subtraction operation. The outcome of A-operation is either a constant to be realized or an intermediate constant. Thus, shift and add operations are enough to realize all constant multiplications.

MCMs in the proposed architecture are designed such that critical path comprises of two adders only. This can be treated as area optimized MCM with constrained delay problem [116]. Next, MCMs in the proposed architecture are reconfigured by inserting multiplexers so that adders can be shared. Design of MCMs lying in datapath P_4 is shown in Fig. 5.5. The same design approach is used to implement the rest of the MCMs.

5.2.2.2 IB unit sharing

Outputs of the IB are shared to facilitate the reuse of MCM as described above. A portion of the shared IB output in datapath P_3 and P_4 is shown in Table 5.2. After multiplying by four different constant sets, these output lines produce four different

Table 5.1: MCM picked by different datapaths

Datapath	MCMs
P1	M_2, M_{13}, M_5, M_{10}, M_3, M_{12}, M_6, M_9
P2	M_1, M_{14}, M_4, M_{11}
P3	M_0, M_{15}
P4	M_7, M_8

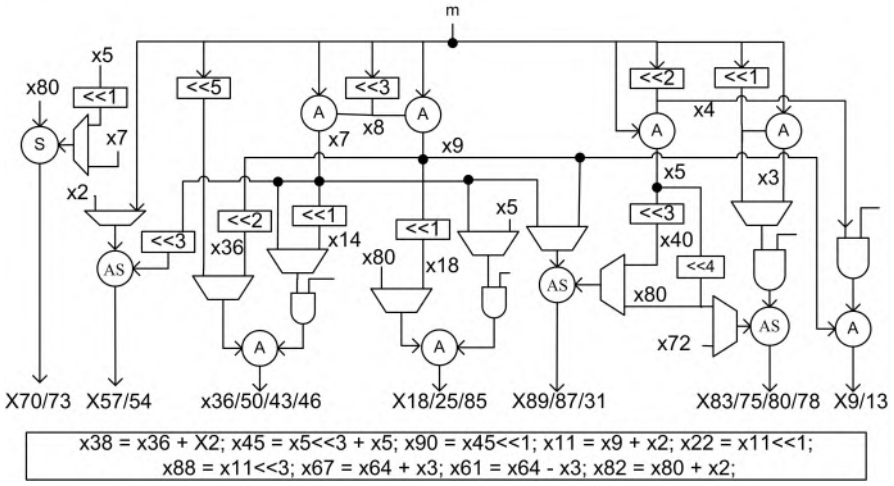

FIGURE 5.5: Architecture of MCM in datapath P_4

DCT coefficients (or intermediate value of coefficients) during different size of DCT calculation as follows:

$$Y_n^N = \pm c_0 \times O_0 \pm c_1 \times O_1 \pm c_2 \times O_2 \pm c_3 \times O_3. \tag{5.4}$$

For example, during 4, 8, 16, and 32 -point DCT calculation $\{c_0, c_1, c_2, c_3\}$ is equal to $\{64, 64, 83, 36\}$, $\{89, 18, 75, 50\}$, $\{87, 25, 80, 43\}$, and $\{88, 22, 78, 46\}$, respectively. Thus, it is possible to produce two 4-point DCT coefficients Y_0^4 and Y_1^4 or a single 8-point DCT coefficient Y_1^8 or intermediate value of 16- and 32- point DCT Y_3^{16} and Y_5^{32}, respectively.

TABLE 5.2: A portion of shared output of IB

DCT size	Output lines			
	O_0	O_1	O_2	O_3
32-point	b_0	b_{15}	b_7	b_8
16-point	b_0	b_7	b_4	b_3
8-point	b_0	b_3	b_1	b_2
4-point	a_0	a_1	b_0	b_1

5.2.2.3 Sharing output adder trees

Although different size of DCT operations are performed, the same adder tree is used to compute various coefficients. Generally, separate adders and subtractors are employed in the output adder trees. However, those are replaced by the add-sub units in the proposed architecture and control signals are generated to perform either addition or subtraction operation. Due to this, it is possible to share output adder trees also.

5.2.3 Datapath truncation

Using the separable property, 2D DCT of $N \times N$ matrix can be calculated by row and column decomposition method. Fig. 5.6 shows the data flow model for HEVC main profile along with intermediate data depth. This model supports 8-bit pixel depth and including sign information residual data depth is 9 bits. These 9 bits are fed as input to the DCT core and finally, 16-bit outputs are produced. It is clear from Fig. 5.6 that after the first level of transform data depth is $log_2N + 15$ bits. However, only 16 bits of intermediate data are stored in the transpose memory. This entails $log_2N - 1$ least significant bits of the data must be truncated. Next, the second level of transform is performed on the data stored in the transpose memory. This operation produces output data whose depth is $log_2N + 22$ and 16-bit outputs are generated by truncating $log_2N + 6$ number of least significant bits. The issue of the different size of bit truncation arises because the first- and second- level DCT input depths are different. Maintaining equal input data depth is a solution to this problem. Such a modified folded 2D DCT architecture is presented in Fig. 5.7. Here, input data are shifted left by 7 bits so that 16-bit data depth is achieved and

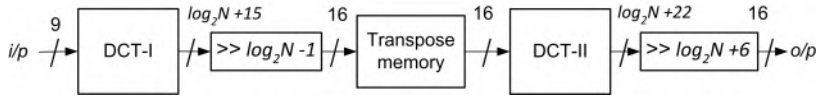

FIGURE 5.6: Data flow model for 2D DCT in HEVC main profile

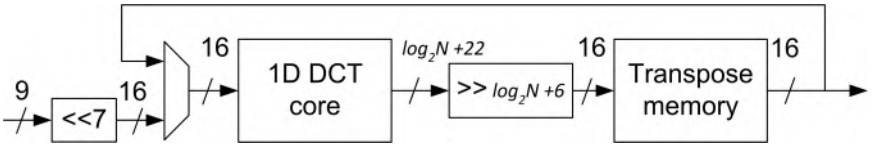

FIGURE 5.7: Modified folded 2D DCT architecture

then it is fed to the DCT core. Next, $log_2N + 6$ number of least significant bits are truncated. This method also produces the same coding efficiency.

The amount of bit truncation and the input data depth of adders vary depending on the DCT size. In the proposed architecture, the bit depth of intermediate data is pruned so that required amount of bit truncation can be achieved with fixed adder size. Datapath truncation for 8-point DCT is shown in Fig. 5.8. Here, 17-bit outputs of IB unit are fed to the MCM as depicted in Fig. 5.8(a). MCMs are designed by right shift operations instead of left shift. Consequently, input bus depth for entire adder structure remains the same and 7 least significant bits also get truncated. Fig. 5.8(b) depicts architecture of MCM-0 and MCM-3 using this technique. Further, in the output adder tree one bit is truncated after each addition, as shown in Fig. 5.8(c). In this way, total $log_2N - 1$ least significant bits are truncated in the output adder tree unit and overall $log_2N - 1 + 7$ bit truncation is achieved without compromising with the expected results.

5.2.4 Transpose Memory (TM)

A 2D DCT architecture requires TM to store intermediate results. A register array is used as TM in [68] because it is highly flexible. However, it is not suitable for the

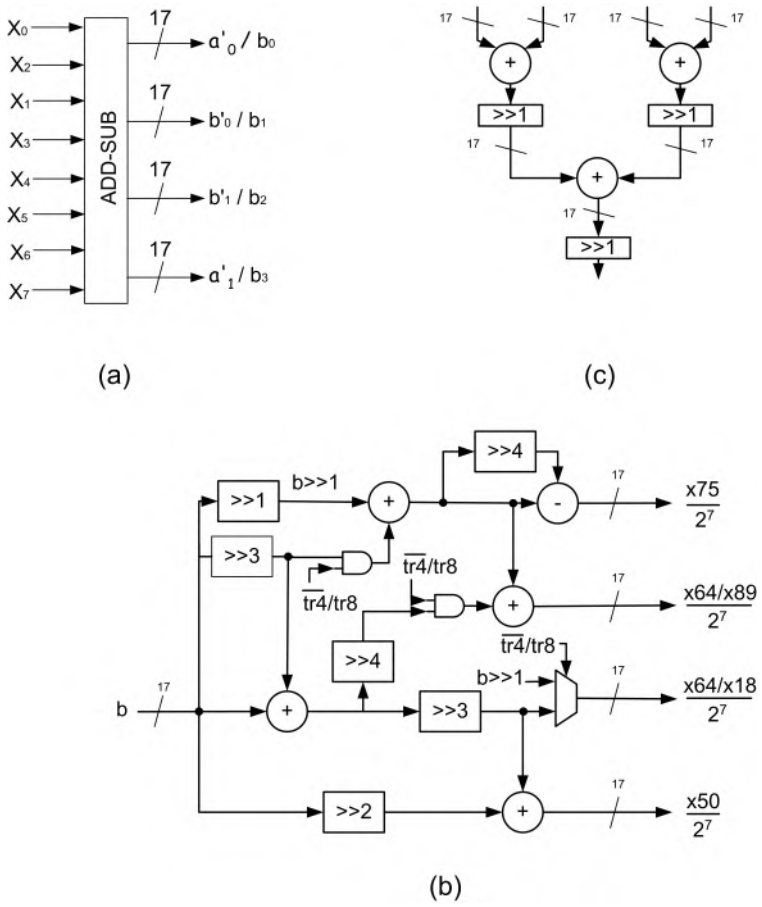

FIGURE 5.8: Datapath truncation example (a) IB (b) MCM-0 and MCM-3 (c) Output adder tree

large size of memory as it is not power and area efficient [117]. Therefore, a single port SRAM based TM is used in the proposed architecture.

The proposed architecture requires N clock cycles to calculate 1D DCT and in this duration, it produces k (i.e. $\frac{32}{N}$) matrices of size $N \times N$. Therefore, a TM with a bandwidth of 32 samples per clock cycle is required. Here, 32 SRAM banks are used and each of them either accepts or emits one sample per clock. Further, each bank contains 32 locations to store intermediate data. A diagonal mapping technique [117] is used while accessing data from them. According to this mapping technique, intermediate results are stored in a diagonal direction as shown in Fig. 5.9(a). It is clear from the figure that element (m, n) of ith matrix is placed in mth address of

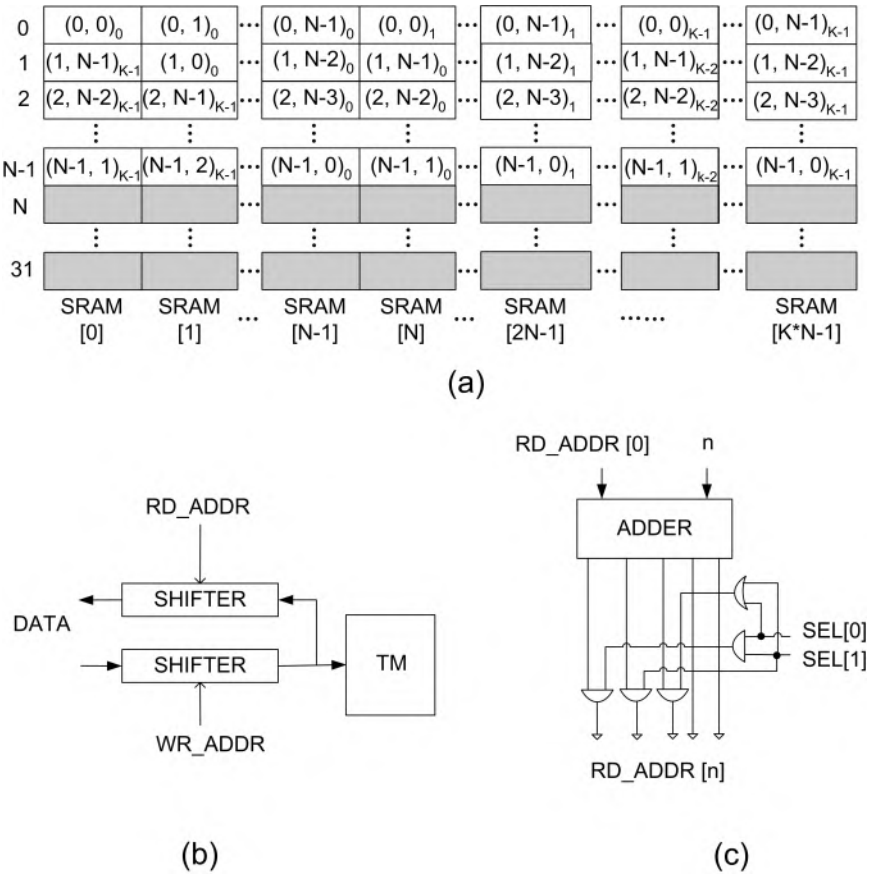

FIGURE 5.9: Transpose memory (a) Data mapping (b) Read and write method (c) Read address generation

$(s \ mod \ 32)$th SRAM bank where

$$s = N \times (i-1) + m + n, \qquad \text{for } 0 \le m, n \le (N-1) \text{ and } 1 \le i \le k. \tag{5.5}$$

This type of mapping is achieved by shifting data before write as well as after read operations, as shown in Fig. 5.9(b). The number of shifts during these operations depends on read and write addresses, respectively.

While reading data 32 SRAM locations are accessed concurrently and one data is read from each bank. Read address for nth SRAM bank can be generated as follows:

$$READ\_ADDR[n] = (BASE\_ADDR + n) \bmod N, \quad \text{for } 0 \leq BASE\_ADDR \leq (N-1). \quad (5.6)$$

Read address from SRAM[0] is treated as a base address. For the first read operation, the base address is zero and it is decremented in the subsequent clock cycles. Fig. 5.9(c) shows the circuit diagram for nth read address generation.

5.3 Results and Discussion

5.3.1 Coding Performance

The proposed architecture uses right shift operations to realize MCMs. As a result, datapath width of the architecture is reduced at the cost of coding performance. To measure the impact of the datapath pruning, we have compared the performance of the architecture with that of the reference software for HEVC [12]. For this performance comparison, HM-16.15 software is used with common test conditions as described in [103]. Input sequences from five different classes are picked. These are PeopleOnStreet 2560×1600, ParkScene 1920×1080, BasketballDrillText 832×480, BasketballPass 416×240 and Johnny 1280×720. PSNR variation is calculated using the standard BD-rate approach [88] with four different quantization parameters (i.e., 22, 27, 32 and 37). The performance comparison results are presented in Table 5.3. Note that the negative PSNR difference represents coding loss compared to that of the reference software. It is evident from Table 5.3 that the maximum PSNR

TABLE 5.3: PSNR difference of the proposed architecture to HM-16.15

	AI			LD			RA		
	Y	C_B	C_R	Y	C_B	C_R	Y	C_B	C_R
A	-0.02	-0.04	-0.03	-	-	-	-0.01	-0.01	-0.01
B	-0.03	-0.05	-0.04	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
C	-0.02	-0.01	-0.02	-0.05	+0.01	-0.02	-0.08	-0.04	-0.02
D	-0.04	-0.05	-0.05	-0.04	-0.08	-0.06	-0.01	-0.03	-0.04
E	-0.08	-0.07	-0.08	-0.01	-0.02	-0.03	-	-	-

variation of the luma component is less than 0.1. Therefore, it suggests that the proposed method of datapath pruning produces an insignificant impact on the coding performance of the encoder.

5.3.2 Implementation Results

In this Section, the proposed architecture is implemented on FPGA as well as ASIC platforms and the implementation results are compared with the existing architectures.

5.3.2.1 FPGA implementation

The proposed 4, 8, 16, and 32 -point 1D DCT architectures are coded in Verilog RTL. All the source codes are then synthesized using Xilinx Vivado 2016.2 to implement on Virtex-7 FPGA platform. For comparison purpose, we have also implemented reference algorithm as proposed in HM software and detailed synthesis report is presented in Table 5.4. Both the architectures use 16-bit input bus. To constrain the speed of operation, registers are used at the input as well as output side. Vector less power estimation is done at 50 MHz clock frequency. Processing time is calculated by reducing positive time slack from the constrained time and the maximum operating frequency is obtained. Since static power, IO power and clock power are not directly

Table 5.4: Synthesis results for 1D DCT architectures

Architecture	Size	LUT	SLICE	Time (ns)	Power (mW)	Throughput (samples / clk)	Area-Delay product	Power reduction	Area-Delay product reduction
Reference	4	300	84	7.046	3	4	2114	-	-
	8	1348	388	7.937	18	4~8	10699	-	-
	16	4943	1428	9.314	68	4~16	46039	-	-
	32	18154	5183	10.589	255	4~32	192233	-	-
Proposed	4	250	76	7.181	3	4	1796	0%	15%
	8	963	298	8.015	12	8	7718	33.3%	27.9%
	16	3754	1171	9.024	40	16	33876	41.2%	26.4%
	32	13680	4165	9.559	126	32	130767	50.6%	32%

Table 5.5: Comparison with other 1D DCT architectures implemented on FPGA platform

	LUT/ALUT	Reg./FF	Speed (MHz)	Throughput (Msps)	FoM	Relative FoM
[83]	13517	-	43.6	1395.8	103.3	0.42
[81]	21568	7309	279.4	2200	102	0.42
Proposed	13680	-	104.6	3347.6	244.7	1

related to the complexity of the design; those are excluded during power calculation.

The motivation of the proposed architecture is to maintain constant throughput irrespective of the TU size along with reduced area overhead. Some architectures [73, 74] have been already proposed which maintain constant throughput irrespective of TU size. However, they use approximate DCT for video compression. Therefore, their coding performance degrades significantly. On the other hand, the proposed architecture uses integer DCT for video compression and its coding performance is almost similar to that of the HM software.

It is clear from Table 5.4 that the proposed architecture requires less number of LUTs as compared to that of the reference architecture. Its processing time requirement and power consumption are also less. It is because the reference architecture uses multipliers to calculate DCT. Additionally, intermediate data bus width is large and finally, 16-bit result is obtained by truncating the output bus. However, only 17-bit intermediate data is used in the proposed architecture and the same result is obtained by truncating a single bit from the output bus. Further, shift and add approach is used to implement multiplication. Consequently, area-delay product and power consumption of the proposed 32-point architecture is 32% and 50.6% lower than that of the reference architecture, respectively. Moreover, the throughput of the proposed N-point architecture is constant at N samples per clock. On the contrary, the throughput of an N-point architecture using reference algorithm reduces during lower order DCT computation and it varies from 4 to N samples per clock.

The performance of the proposed architecture is compared with the other existing DCT/IDCT architectures implemented on FPGA platform. These comparison results are presented in Table 5.5. From this table, it is clear that LUT utilization of the proposed architecture is almost equal to that of [83]. However, the processing speed of the proposed architecture is more than 2.5 times that of [83]. Architecture

in [81] operates at very high frequency but, its throughput varies depending on the type of DCT applied. On the other hand, throughput of the proposed architecture is constant irrespective of the DCT size and it is much higher in comparison to the other architectures. To determine the overall efficiency and for a fair comparison, figure of merit (FoM) is calculated for all the designs as follows:

$$FoM = \frac{Total\ samples\ processed\ per\ second}{Area \times 10^3}, \tag{5.7}$$

where the area is proportional to the number of LUTs required for the design. Relative FoM is calculated by normalizing FoM with respect to the maximum FoM. It is clear that FoM of the proposed architecture is the best among those listed in the table.

5.3.2.2 ASIC implementation

The source code of the proposed 1D DCT architecture is also implemented on ASIC platform using 90 nm standard cell library [104]. These results are also compared with some of the existing DCT/IDCT architectures and are presented in Table 5.6. Total area of the proposed architecture is divided by the area of 2-input NAND gate to estimate the gate count and power consumption is estimated by constraining the frequency at 100 MHz.

It is observed that the hardware cost of the proposed architecture is 79.2K logic gates and the maximum operating frequency is 157 MHz. FoM and relative FoM are calculated using (5.7), where the area has been replaced with gate count for this purpose. FoM and relative FoM reveal that the performance of the proposed architecture is better than all but one [80] design listed in the Table 5.6. This is because design [80] is a pipelined architecture and it is implemented on advanced

Table 5.6: Comparison with other 1D DCT architectures implemented on ASIC platform

	[68]	[80]	[84]	[76]	[71]	[85]	Proposed
Technology	90 nm	65 nm	130 nm	90 nm	90 nm	90 nm	90 nm
Gate counts	131 K	115.8 K	93 K	63.8 K	163 K	63.8 K	79.2 K
Max. Freq (MHz)	187	476	191	270	250	312	157
Power (mW)	23.17	6.12	-	-	35.38	-	15.8
Throughput (samples / clock)	32	32	4	8	25.7	4	32
FoM	45.64	131.5	8.22	33.85	39.4	19.56	63.4
Relative FoM	0.72	2.07	0.13	0.53	0.62	0.31	1

Table 5.7: Computational complexity of 1D DCT architectures

DCT size		4	8	16	32
Reference	MUL	4	22	86	342
	ADD	8	28	100	372
	SHIFT	2	4	8	16
Reusable [68]	ADD	14	64	264	1024
	SHIFT	10	40	136	464
Proposed	ADD	14	52	200	712
	SHIFT	10	42	146	395

technology node. Table 5.6 entails that gate count requirement of the proposed architecture is higher than that of [76] and [85]. Similarly, the speed of the architectures [84], [76] and [85] is much higher as compared to that of the proposed architecture. However, design in [84] and [85] can process only four pixels in a single clock cycle. Similarly, IDCT architecture in [76] supports all size of transforms used in HEVC, but its throughput depends on the transform size. Similarly, IDCT architecture in [76] supports all size of transforms used in HEVC, but its throughput depends on the transform size. It requires 11 cycles to process one row of a 32-point transform in the worst case. Therefore, the effective throughput of those architectures is less. As a consequence, to perform 2D DCT of a TU, they require more number of clock cycles as compared to the proposed architecture. Hence, none of them is suitable to process 30 frames of 4:2:0 format 8K UHD video in a second. The design in [71] also tries to maintain constant throughput irrespective of the DCT size with the use of lifting algorithm. As a result, its operating frequency is high. However, its latency is high and it depends on the number of pipeline stages. Therefore, the effective throughput of the design [71] is less than that of the proposed architecture. Additionally, it requires extra hardware to reduce the power consumption which increases area overhead. The reusable architecture [68] and the proposed design are capable of producing 32 samples per clock and their throughput remains constant irrespective of the DCT size. However, the area of the design [68] is higher because it uses an extra $C_{\frac{N}{2}}$ unit with each N-point DCT module to obtain constant throughput. On the other hand, the proposed architecture uses hardware sharing method to maintain constant throughput. Therefore, hardware complexity of the proposed architecture, in terms of number of adders and shifters, is less as compared to that of [68]. Table 5.7 presents a comparison of computational complexity with other 1D DCT architectures. The proposed architecture and reusable architecture [68] avoid the use of multiplication and MCMs are used for this purpose. However, the number

TABLE 5.8: Comparison of folded 2D DCT architectures implemented on ASIC platform

Architecture	N	Technology	Area (Gates)	Memory (bits)	Speed (MHz)	Througput (pel)	Power (mW)
[54]	8	45 nm	86 K	-	806	6.45 G	600
[56]	8	45 nm	74 K	-	763	6.1 G	500
[58]	8	45 nm	78 K	-	809	6.47 G	550
[62]	8	45 nm	74 K	-	849	6.79 G	550
[118]	8	45 nm	74 K	-	851	6.8 G	550
[68]	4~32	90 nm	208 K	-	187	2.99 G	40
[70]	4~32	90 nm	149 K	-	150	253 M	-
[71]	4~32	90 nm	243 K	-	250	3.21 G	52
[119]	4~32	90 nm	383 K	16 K	311	9.72 M	35.4
Proposed	4~32	90 nm	87 K	16 K	146.4	2.3 G	28

of adders required in the proposed architecture is less as compared to the reusable architecture [68] because hardware sharing method is used.

We have also coded the proposed folded architecture for 2D DCT calculation in Verilog. It is synthesized using 90 nm standard cell library and the comparison results with some of the architectures are presented in Table 5.8. The architectures in [54, 56, 58, 62, 118] are able to compute 8-point DCT and do not support variable TU size requirement of HEVC. Although throughput of those architectures is very high, they consume huge power. Architectures in [70, 71, 119] use pipeline design strategy to improve the throughput. On the contrary, the proposed design uses resource sharing technique for this purpose. Consequently, significant reduction in gate count and power consumption is observed. Moreover, throughput of the proposed architecture is also comparable to the design in [71]. The proposed 2D DCT architecture uses single port SRAM banks as transpose memory and it takes $(2N+1)$ cycles to calculate 2D DCT of $\frac{32}{N}$ blocks of size $N \times N$. To support 8K UHD video at 4:2:0 YUV format with 30 frames per second any architecture must be able to process 1.5 G pixels per second. However, the proposed 2D DCT architecture can process maximum 2.3 G pixel per second. Therefore, in a second this architecture can process at most 185 frames and 46 frames of $4K$ and $8K$ size UHD video, respectively.

5.4 Summary

In this chapter throughput of the integer DCT architecture has been increased to improve the processing time in all quadtree depth. It is observed that lower order DCTs are used frequently than higher order DCTs during video encoding in HEVC standard. However, throughput of the transform core reduces during lower order

DCT computation. Therefore, a resource sharing scheme is used in the proposed architecture so that multiple numbers of lower order DCTs are computed concurrently. The proposed 1D DCT architecture can process 32 samples/clock irrespective of the transform size. Maximum operating frequency of the proposed 1D DCT architecture is 104.6 MHz when implemented on Virtex-7 FPGA and it can process 3347.6 M samples per second. CMOS 90 nm ASIC implementation of the same design requires 79.2K logic gates and it operates at 157 MHz. The proposed modified 2D DCT architecture requires $2N + 1$ clock cycles to process $\frac{32}{N}$ blocks of $N \times N$ pixels. The proposed folded 2D DCT model requires 87K gates and its throughput is 2.3 G pixels per second. The maximum frequency of operation of this architecture is 146.4 MHz and power consumption is 28 mW. Therefore, the proposed 2D DCT architecture can process at most 185 frames and 46 frames of 4K and 8K ultra high definition (UHD) video, respectively.

—-