Recursive algorithm, architectures and FPGA implementation of the 2-D DCT

Shaofeng An and Chunyan Wang
Department of Electrical and Computer Engineering
Concordia University
1455 Maisonneuve Blvd. West, Montreal, Canada H3G 1M8
{sh_an, chunyan}@ece.concordia.ca

Abstract: In this paper, a new recursive algorithm and two types of circuit architectures are presented for the computation of the two dimensional discrete cosine transform (2-D DCT). The new algorithm permits to compute the 2-D DCT by a simple procedure of the 1-D recursive calculations involving only *cosine* coefficients. The recursive kernel for the proposed algorithm contains a small number of operations. Also, it requires a smaller number of pre-computed data compared to many of existing algorithms in the same category. The kernel can be easily implemented in a simple circuit block with a short critical delay path. In order to evaluate the performance improvement resulting from the new algorithm, an architecture for the 2-D DCT designed by direct mapping from the computation structure of the proposed algorithm has been implemented in a FPGA board. The results show that the reduction of the hardware consumption can easily reach 25% and the clock frequency can increase 17% compared to a system implementing a recently reported 2-D DCT recursive algorithm. For a further reduction of the hardware, another architecture has been proposed for the same 2-D DCT computation. Using one recursive computation block to perform different functions, this architecture needs only approximately one half of the hardware that is required in the first architecture, which has been confirmed by a FPGA implementation.

Index Terms — Discrete Cosine Transform (DCT), 2-D DCT, recursive algorithms, architecture, FPGA implementation.

1 Introduction

The discrete cosine transform (DCT) [1] is widely used in the area of signal and image processing. In particular, in case of images with high correlation coefficients, DCT based coding can result in a good performance [2]. The 2-D DCT is used in many image/video coding compression standards such as JPEG [3], ITU-T H.261 [4], and MPEG [5]. To facilitate real-time implementations, the development of efficient algorithms for computing the 2-D DCT is of great interest.

The computation of the 2-D DCT requires a large number of operations with a huge amount of data. This computation is usually decomposed into 1-D DCT/DST ones. Some of the most commonly used algorithms are based on the row-column decomposition scheme because of its simplicity of the calculation procedure resulting from the simple row-by-row and column-by-column 1-D operations [6] - [9]. Aiming at reducing the number of calculation cycles for fast processing, some other algorithms take less strait-forward decomposition methods than the row-column ones [10] - [13]. In these cases, the 2-D DCT computation is reformulated, by means of, e.g., conversions of variables, into terms of 1-D DCTs and/or DSTs. It is important that the reformulated computation keeps its regularity and modularity to facilitate the implementation.

Some of the 2-D DCT algorithms are designed for parallel processing for high speed applications [14] - [16]. However, in many applications, low cost and small circuit volume may be demanded. A 2-D DCT algorithm featuring highly regular calculations can be implemented in a small number of circuit blocks for it to be performed in a recursive manner [17] - [20], which may help to achieve a low hardware cost. In general, such a recursive approach is related to a slow process, as the computation task is completed by means of cycle-by-cycle calculations. However, the penalty of such calculation

cycles on the processing speed may not be sever if i) a good decomposition method is used to minimize the number of calculation cycles, and ii) an optimized recursive kernel is designed to minimize the time required for each cycle.

The objective of the work presented in this paper is to develop a system for the 2-D DCT computation with a very low circuit cost and reasonably fast speed. For this purpose, a recursive structure is designed. The computation algorithm proposed in this paper is based on that reported in [20] as it is one of the well appreciated existing recursive algorithms because of its small number of computation cycles.

This paper is organized as follows. The description of the proposed algorithm is founded in Section 2. Section 3 is dedicated to the presentation of the two architectures implementing the proposed algorithm, including the structure of the recursive kernel and the circuit architectures for the recursive computation. Both architectures have been implemented using FPGA technology. The results are also presented in Section 3.

2 Proposed recursive algorithm for the 2-D DCT

2.1 Background

For a set of 2-D data $x(n_1, n_2)$ with $0 \le n_1 \le N-1$ and $0 \le n_2 \le N-1$, the 2-D DCT is defined as

$$X(k_1, k_2) = \frac{2}{N} u(k_1) u(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \times \cos \frac{2\pi (2n_1 + 1) k_1}{4N} \cos \frac{2\pi (2n_2 + 1) k_2}{4N} \quad (1)$$

where $k_1, k_2 = 0, 1, \ldots, N-1$, $u(k) = 2^{(-1/2)}$ for $k = 0$, and $u(k) = 1$ for $k \neq 0$. The 2-D DCT is often reformulated as the sum of 1-D DCTs and/or 1-D DSTs. In [20], assuming that $N = r^p$ and any integer could be expressed as $(d\, r^p)$, with r being a prime integer, d relatively prime to r and $d < N$, k_1

and k_2 are expressed as $k_1 = d_1 r^{p1}$ and $k_2 = d_2 r^{p2}$, and the computation of the 2-D DCT is divided into two cases as follows.

Case 1: k_1 or k_2 is prime to N. Let $p = min\ \{p_1, p_2\}$, $\omega_1 = d_1$, $\omega_2 = d_2$ and $M = N$.

Case 2: $k_1 \neq 0$ and $k_2 \neq 0$, k_1, k_2 and N have a common divisor r^p where $p = min\ \{p_1, p_2\}$, $\omega_1 = d_1 r^{-p1}$, $\omega_2 = d_2 r^{-p2}$ and $M = N r^{-p}$.

In each of these two cases, (1) can be written as

$$X(\omega_1 r^p, \omega_2 r^p) = \frac{1}{2}\left\{\begin{array}{l} \sum_{m_1=0}^{M-1} x_a(m_1)\left[\cos(m_1\pi/M)\cos((\omega_1+\omega_2)\pi/2M) - \sin(m_1\pi/M)\sin((\omega_1+\omega_2)\pi/2M)\right] \\ + \sum_{m_2=0}^{M-1} x_s(m_2)\left[\cos(m_2\pi/M)\cos((\omega_1-\omega_2)\pi/2M) - \sin(m_2\pi/M)\sin((\omega_1-\omega_2)\pi/2M)\right] \end{array}\right\} \quad (2)$$

where $\omega_1 = d_1$, $\omega_2 = d_2$, $M = N$ for case 1, and $\omega_1 = d_1 r^{-p1}$, $\omega_2 = d_2 r^{-p2}$, $M = N r^{-p}$ for case 2. $m_1 = \omega_1 n_1 + \omega_2 n_2\ mod\ M$, and $m_2 = \omega_1 n_1 - \omega_2 n_2 + 2M^2\ mod\ M$. The computation of pre-addition $x_a(m_1)$ and $x_s(m_2)$ can be found in [20].

In the algorithm presented above, the computation of the 2-D DCT can be reformulated into terms comprising 1-D DCTs and 1-D DSTs. The recursive kernel for the computation is shown in Fig. 1. It can also be presented using a more compact format as illustrated in Fig. 2. The diagram of the structure of the recursive algorithm in [20] is shown in Fig. 3. In this approach, each computation cycle requires six multiplications with the DST and DCT coefficients as shown in Fig. 2. It would be desirable to further simplify the computation so as to reduce the number of multiplications in the recursive kernel while removing the DST calculation.

2.2 Proposed recursive algorithm

Based on the method described in Section 2.1, our work aims at developing a computation algorithm for the 2-D DCT using 1-D DCT modules with a reduced number of multiplications. The equation (2)

for the 2-D DCT computation involves multiple *sine* and *cosine* terms. Totally six multiplications are required in each cycle. A further simplification of the computation is to reduce the number of multiplications and to make the computation to contain only *cosine* (or *sine*) terms.

Observing the equation (2), it is easy to see that, by using trigonometry identities $cos(u + v) = cosu\ cosv - sinu\ cosv$, the 2-D DCT computation shown in (2) can be expressed as [21].

$$X(\omega_1 r^p, \omega_2 r^p) = \frac{1}{2}\left\{\sum_{m_1=0}^{M-1} x_a(m_1)\cos\left((m_1 + \frac{\omega_1+\omega_2}{2})\pi/M\right) + \sum_{m_2=0}^{M-1} x_s(m_2)\cos\left((m_2 + \frac{\omega_1-\omega_2}{2})\pi/M\right)\right\} \quad (3)$$

Comparing (3) and (2), one can find that the number of multiplications used in each cycle of the recursive computation in (3) is evidently smaller than that in (2). Thus, the recursive kernel for the computation of (3) can be made much simpler than that of (2). Moreover, as (3) involves only 1-D DCT terms, the generation of 1-D DST is not needed, which permits a significant simplification of the pre-computation of the 2-D DCT.

Equation (3) can be written as the sum of X_{ac1} and X_{ac2} defined as

$$X_{ac_1}(\omega_1, \omega_2) = \sum_{m_1=0}^{M-1} x_a(m_1)\cos\left((m_1 + \frac{\omega_1+\omega_2}{2})\pi/M\right) \quad (4)$$

$$X_{ac_2}(\omega_1, \omega_2) = \sum_{m_2=0}^{M-1} x_s(m_2)\cos\left((m_2 + \frac{\omega_1-\omega_2}{2})\pi/M\right) \quad (5)$$

Observing (4) and (5), one can find that they differ form each other only in the angles $(\omega_1 + \omega_2)/2$ and $(\omega_1 - \omega_2)/2$. It is thus possible to use the same structure to implement the computation of X_{ac1} or X_{ac2}. Also, the computation procedure should be designed in such a way that this structure involves the minimum number of multipliers and requires the shortest delay for the computation.

Assuming that $\omega = (\omega_1 \pm \omega_2)/2$, $m' = M - 1 - m$, and $x_a(m)$ or $x_s(m)$ is generalized as $x(m)$, (4) or (5) can be expressed as

$$X_{ac}(\omega_1,\omega_2)=\sum_{m=0}^{M-1}x(m)\cos\left(\frac{m+\omega}{M}\pi\right)=\sum_{m'=0}^{M-1}x(M-1-m')\cos\left(\pi-\frac{1+m'-\omega}{M}\pi\right)$$
$$=-\sum_{m'=0}^{M-1}x(M-1-m')\cos\left(\frac{1+m'-\omega}{M}\pi\right) \quad (6)$$

Making $Y(\omega_1, \omega_2) = - X_{ac}(\omega_1, \omega_2)$, $j = M - 1$ and $\theta = \pi/M$, we have

$$Y(\omega_1,\omega_2)=-X_{ac}(\omega_1,\omega_2)=\sum_{m'=0}^{M-1}x(M-1-m')\cos\left(\frac{1+m'-\omega}{M}\pi\right)=\sum_{m'=0}^{j}x(j-m')\cos(1+m'-\omega)\theta \quad (7)$$

As $cos(m' + 1)\theta = 2cos\theta cosm'\theta - cos(m' - 1)\theta$, (7) can be rewritten as

$$\begin{aligned}Y(\omega_1,\omega_2)&=\sum_{m'=0}^{j}x(j-m')\cos(1+m'-\omega)\theta=\sum_{m'=0}^{j}x(j-m')\left[2\cos\theta\cos(m'-\omega)\theta-\cos(m'-\omega-1)\theta\right]\\&=\left\{2\cos\theta\left[\cos(\omega\theta)x(j)+Y_{j-1}(\omega_1,\omega_2)\right]-\left[\cos(\omega+1)\theta x(j)+\cos(\omega\theta)x(j-1)+Y_{j-2}(\omega_1,\omega_2)\right]\right\}\\&=\left\{\left[2\cos\theta\cos\omega\theta-\cos(\omega+1)\theta\right]x(j)-\cos\omega\theta x(j-1)+2\cos\theta Y_{j-1}(\omega_1,\omega_2)-Y_{j-2}(\omega_1,\omega_2)\right\}\\&=\left[\cos(\omega-1)\theta x(j)-\cos\omega\theta x(j-1)+2\cos\theta Y_{j-1}(\omega_1,\omega_2)-Y_{j-2}(\omega_1,\omega_2)\right]\end{aligned} \quad (8)$$

The transform function of the system for (8) is given as

$$\frac{Y\left((\omega_1,\omega_2),z\right)}{X\left(z\right)}=\frac{\cos(\omega-1)\theta-\cos\omega\theta Z^{-1}}{1-2\cos\theta Z^{-1}+Z^{-2}} \quad (9)$$

It should be noted that (9) is applicable for both $X_{ac1}(\omega_1, \omega_2)$ and $X_{ac2}(\omega_1, \omega_2)$ as $\omega = (\omega_1 \pm \omega_2)/2$.

Based on (9), we propose a recursive computation kernel as shown in Fig. 4. The structure shown in Fig. 5 includes the proposed kernel and is for the same 2-D DCT computation as that of Fig. 3. Comparing the kernel shown in Fig. 4 with that in Fig. 2, one can see that the former needs only *cosine* coefficients, and involves four multiplications and three additions, whereas the latter requires both *cosine* and *sine* coefficients and employs six multiplications and four additions. It can be expected that the proposed algorithm can be implemented in a simpler circuit structure, with potentially shorter clock cycle-time.

To verify the equivalence of the computation expressed in the recursive kernel illustrated in Fig. 2 and that in Fig. 4, simulations using Simulink have been done and identical results have been obtained [21], conforming that the kernels can replace each other for the same computation. The option of

computing the 2-D DCT by the proposed recursive operation illustrated in Fig.4 can lead to an effective reduction of number of multiplications and additions.

It should be mentioned that, similar to the recursive kernel reported in [20], the proposed one can be used not only for the DCT, bet also in the computations of the IDCT, the DST and the IDST, by means of adjusting the inputs and pre-addition procedures.

3 Architectures and the FPGA implementations

As described in the preceding section, the proposed algorithm of the 2-D DCT is based on the computations defined by (3), and a recursive kernel shown in Fig. 4 involving only the 1-D DCT computation has been developed. We propose a circuit block for the recursive kernel and two versions of VLSI architecture to implement the algorithm, aiming at improving the hardware usage and operation speed. Both versions of architectures are implemented in FPGA boards for performance evaluation. In this section, a 1-D DCT circuit block of the recursive kernel and the two architectures are presented, so are the results of the FPGA implementation.

3.1 Circuit block of the recursive kernel

The computation of the proposed recursive kernel, shown in Fig. 4, can be easily mapped into a simple circuit block as shown in Fig. 6. This circuit does not need *sine* coefficients and it is, in fact, a 1-D DCT block. Using the same mapping, another block for the recursive kernel [20] is also built, illustrated in Fig. 7, for the comparison purpose.

Comparing the structures of the two blocks shown in Figs. 6 and 7, one can have the following observations:

- It is confirmed that the block shown in Fig. 6 requires four multipliers and three adders, instead of six multipliers and four adders in that in Fig. 7. Thus, a significant reduction of circuit complexity should be expected in the hardware implementation.
- Besides the inputs of x_a and x_s generated by the pre-computation modules, both blocks receive other pre-computed inputs. The block shown in Fig. 7 needs six such inputs, namely *a, b, c, d, e* and *f*, whereas in that of Fig. 6 only four are needed. Therefore, the pre-computation operations required in the system using the block of Fig. 6 can be made much simpler than that of Fig. 7.
- The length of the most critical delay path in a block determines the required duration of the clock cycle. One can easily see that there are three multipliers in the critical path of the block of Fig. 7, and only two in that of Fig. 6. The delay in the latter is obviously much shorter than that in the former. Also, taking the number of adders in the critical path of each of the two blocks into consideration, one can expect that the delay of the circuit block of the proposed recursive kernel can be made at least 33% shorter than that of Fig. 7. The proposed algorithm can, therefore, lead to a significant increase of the clock frequency.

Having a smaller number of operators such as multipliers and adders and fewer input coefficients, the circuit block for the proposed recursive kernels can perform a smaller number of operations for the same computation as that shown in Fig. 7. It can thus be implemented with a smaller number of basic calculation units and shorter delay path to improve both hardware efficiency and processing speed. The VLSI architectures for the 2-D DCT with the proposed block shown in Fig. 6 are presented in the following sub-sections.

3.2 Architecture -1 for the proposed algorithm

As mentioned in the previous sections, the 2-D DCT computation can be implemented by using the proposed circuit block of the new recursive kernel shown in Fig. 6. With this block, the computation structure shown in Fig. 5 can be easily mapped into an architecture illustrated in Fig. 8. In this architecture, two identical circuit blocks operate in parallel. However, it should be noted that one block receives the inputs $\alpha = cos((\omega_1+\omega_2)/2\text{-}1)\pi/M$ and $\gamma = cos((\omega_1+\omega_2)\pi/2M)$, while the inputs $\beta = cos((\omega_1-\omega_2)/2\text{-}1)\pi/M$ and $\delta = cos((\omega_1-\omega_2)\pi/2M)$ are applied to the other block. Hence, the former produces X_{ac1} and the latter X_{ac2}. The final output signal $X(k_1, k_2)$ is generated by an addition of the two outputs of the blocks.

Using a similar direct mapping, a circuit architecture implementing the algorithm of [20] is obtained, as illustrated in Fig. 9. It includes two identical circuit blocks shown in Fig. 7.

The architectures shown in Figs. 8 and 9 can be easily implemented with FPGA technology. To this end, one should first decide the target precision of the 2-D DCT computation. There is no limit for the precision in each of the two architectures except the number of bits of the modules employed, providing that the clock frequency is appropriate. However, a larger number of bits requires more devices in the modules, i.e. larger or more expensive FPGA boards. Most of 2-D DCT applications use inputs of 12-bit floating-point data [9] [22]. In the implementation of the two architectures, the inputs and outputs are also of 12-bit floating-point signals, of which the first bit is the sign, the next three bits are of the exponent, and the other eight bits are for the significand.

In this implementation, VHDL is used for hardware description and the Mentor Graphics Precision RTL for the logic synthesis. The netlist files are of EDIF (Electronic Design Interface Format), and Xilinx ISE is used for the FPGA simulation. The FPGA boards used are Virtex-II Pro

Platform of xc2vp7, featuring the volume of 4928 slices and the maximum clock frequency of 500 MHz. The input data applied to the two architectures are the same, and the same output data are obtained and verified to be correct. The hardware simulation results are presented in Table I. One can note the following points.

- The architecture of the proposed algorithm requires only 75% of hardware consumption compared to that of [20]. This results from the smaller number of multiplications and additions in the proposed algorithm that involves only 1-D DCT computation.
- Both architectures can be easily implemented in the FPGA board of xc2vp7. Obviously, the hardware utilization of the architecture shown in Fig. 8 is much less than that of Fig. 9, which means that the architecture of Fig. 8 permits the integration of more logic functions in the same board. Furthermore, this architecture can also be implemented in a small and low-cost board such as xc2vp4 that may not be suitable to implement the one of Fig. 9.
- Because of the smaller number of operations in the critical path of the computation, the required clock cycle duration of the architecture shown in Fig. 8 is 7.61 ns, corresponding to the frequency of 131 MHz. It is only 83% of that of [20], which implies a speed improvement resulting from the proposed algorithm. The reduction of the cycle time can be more significant if the algorithm is implemented in a custom-designed integrated circuit.
- The circuit architecture for the proposed algorithm and that of [20] dissipate almost the same amount of power in the FPGA boards. As the former has a simpler structure, when it is implemented in a custom designed circuit, its power dissipation is expected to be lower than that of [20].

TABLE I

FPGA RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 8 AND 9

		Architecture shown in Fig. 9	Architecture shown in Fig. 8
Slice Registers	Number	272	217
	Utilization	2%	2%
Occupied Slices	Number	4,008	2,997
	Utilization	81%	60%
4 Input LUTs	Number	7,710	5,777
	Utilization	78%	58%
Minimum Clock Cycle (ns)		9.103	7.611
Power Consumption (mW)		472	472

The throughput of the architecture shown in Fig. 8 is the same as that in Fig. 9, which is one input sample per clock cycle. Both of the architectures need N clock cycles to produce an output sample if the dimension of the 2-D signal is $N \times N$. By means of the input-folding method, the number cycles may be further reduced in the architecture shown in Fig 9 [20], and this reduction is done at the expense of increasing the hardware consumption.

The simulation using Simulink was done with the input data of fixed-point. The FPGA results are obtained with the floating-point signals. As shown in Table I, the clock cycle can be as high as 7.611 ns. We can conclude that the circuit can operate correctly with a clock frequency of 130 MHz under the condition of the 12-bit floating-point data.

In conclusion, the FPGA results agree with the expected performance improvement of the circuit architecture resulting from the proposed algorithm. The improvement, in terms of hardware consumption and operation speed, has been achieved at no expense of power dissipation.

3.3 Architecture -2 for the proposed algorithm

The architecture shown in Fig. 8 employs two circuit blocks to generate X_{ac1} and X_{ac2}, respectively, for the 2-D DCT computation. The two blocks execute, in fact, the same operation with partially different inputs. It is thus possible to use only one circuit block for both functions of X_{ac1} and X_{ac2}, and the matter is to select the right inputs for the two functions. By examining the architecture shown in Fig. 8, it is easy to see that the calculation of X_{ac1} requires $\alpha = cos((\omega_1+\omega_2)/2\text{-}1)\pi/M$ and $\gamma = cos((\omega_1+\omega_2)\pi/2M)$, while X_{ac2} needs $\beta = cos((\omega_1-\omega_2)/2\text{-}1)\pi/M$, and $\delta = cos((\omega_1-\omega_2)\pi/2M)$. One can use simple multiplexers to select the input signals (α & γ) or (β & δ) in order that the circuit block produces the right output. In this way, the proposed algorithm can be implemented in the circuit using only one circuit block for all the recursive operations, as illustrated in Fig. 10, which can result in a significant reduction of the hardware consumption and enable an even-lower-cost circuit implementation.

In the architecture shown in Fig. 10, the signal inputs via the multiplexers are synchronized with the clock signal, of which each cycle consists of two phases. In the first phase, i.e. CLK = '1', x_a, α and γ are applied to the circuit block and during the second phase, x_s, β and δ are selected to be applied. The two outputs of the circuit block generated during the two phases are summed up to generate the final output signal X (k_1, k_2). In this way, the computation of the 2-D DCT can be realized by only one 1-D DCT block, instead of the two in Fig. 8.

The architecture in Fig. 10 can also be used for other computation tasks if the main circuit block is replaced by another module. In many cases of signal processing, the computation can be decomposed into two parts, one by processing cores and the other by pre-computation blocks. It is possible to use only one processing core for different functions during different phases, while applying different

pre-computed inputs. In this manner, the required hardware can be considerably reduced, which leads to a reduction of the circuit cost in a great scale.

The same method of using a single processing core can be easily applied to simplify the structure implementing the algorithm of [20]. In the architecture shown in Fig. 9, among the inputs *a, b, c, d, e* and *f* of each circuit block, *b* and *e* are not common for the two blocks. By means of multiplexers, the structure in Fig. 9 is converted to that illustrated in Fig, 11. It is evident that the conversion reduces the hardware to one half of that used for Fig. 9, as only one processing block, instead of two, is included in the circuit.

The circuit architectures shown in Figs. 10 and 11 have been implemented in the same kind of FPGA boards as that presented in Section 3.2, i.e. Virtex-II of xc2vp7, under the same conditions of 12-bit floating-point input signals. The FPGA results of the architecture shown in Fig. 10, with the comparison to those of Fig. 8, are presented in Table II to illustrate the difference in hardware consumption between the two architectures that employ the same recursive block and perform the same computation. Table III shows the results of those of Figs. 9 and 11.

TABLE II

FPGA RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 8 AND 10

Structures for the proposed algorithm		Architecture shown in Fig. 8	Architecture shown in Fig. 10
Slice Registers	Number	217	148
	Utilization	2%	1%
Occupied Slices	Number	2,997	1,709
	Utilization	60%	34%
4 input LUTs	Number	5,777	3,241
	Utilization	58%	32%
Minimum Clock Cycle (ns)		7.611	20.916

TABLE III

FPGA RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 9 AND 11

Structures for the algorithm in [20]		Architecture shown in Fig. 9	Architecture shown in Fig. 11
Slice Registers	Number	272	172
	Utilization	2%	1%
Occupied Slices	Number	4,008	2,207
	Utilization	81%	44%
4 input LUTs	Number	7,710	4,203
	Utilization	78%	42%
Minimum Clock Cycle(ns)		9.103	25.592

From the FPGA results shown in Table II and Table III, the following points can be noticed.

- The proposed method of using a single processing core helps to reduce significantly the hardware consumption in all the aspects, including logic gates and memories. In the case of implementing the two different 2-D DCT algorithms, the reduction of the hardware is consistently at a rate of 43%.
- As the result of the reduced hardware requirement, the circuit architectures designed using the proposed method can be easily integrated in a wide range of FPGA boards.
- By using the proposed method, the recursive block used in each of the circuit architectures shown in Figs. 10 and 11 operates to compute X_{ac1} and X_{ac2} successively, not simultaneously, in each clock cycle. Thus the duration of the cycle is expected to be doubled, compared to that of the architectures shown in Figs. 8 and 9. However, Table II or III shows that the required clock cycle of the architecture illustrated in Fig. 10 or 11 is about three times of that in Fig. 8 or 9, which may be due to some redundant structure of the FPGA board.

The FPGA results provide a good confirmation of the significant reduction of hardware consumption by applying the one-processing-core method. This reduction comes with some increase of processing time and is good to use if the speed is not a critical issue. In any case, it provides a good trade-off of hardware and processing time. Also, as one of the experts in the area suggested, if the circuit of the recursive kernel is divided into cascaded stages to make the operations pipelined, the clock frequency can be increased, as its cycle duration will depend on the stage that has longest delay instead of the total delay of the stages. In this case, the synchronization of the Mux and Demux should be updated to suit the higher frequency of the clock.

It should be mentioned that many 2-D DCT architectures found in literature are implemented in custom-designed VLSI. It is thus difficult to compare the implementation of our architectures with them. There are a few examples of the FPGA implementations reported. They are mostly about architectures designed for row-column decomposition. One such example can be found in [23]. In this architecture, the number of devices needed is almost doubled compared to that of the architecture-2, so is its clock duration.

4 Conclusion

In this paper, a recursive algorithm for the 2-D DCT has been proposed and two architectures implementing the proposed algorithm designed. The development of the new algorithm is based on a mathematical reformulation of the 2-D DCT computation. It involves only 1-D DCT calculations, eliminating 1-D DSTs. The recursive kernel of this algorithm requires a very small number of multiplication and addition operations for an easy circuit implementation. A circuit block of the recursive kernel has also been proposed. This block employs only four multipliers and three adders. Moreover, because of a small number of operators in its critical path, the required clock cycle duration

can be made short. Furthermore, with some variations of the pre-additions, the proposed recursive kernel can also be employed to compute the 2-D IDCT, DST and IDST. Based on the proposed recursive block, two architectures for the recursive computation of the 2-D DCT have been designed. One is obtained by a simple mapping of the computation scheme and includes two proposed recursive circuit blocks. The FPGA results show that, the proposed recursive algorithm leads to 25% of the hardware reduction and 17% of the increase of the clock speed. The other architecture, designed using a one-processing-core method, has also been presented. This architecture uses only the single recursive circuit block for the same 2-D DCT computation. It is shown by the FPGA implementation that this architecture needs only 57% of the hardware consumption of that required by the first architecture with some extension of the clock cycle. The FPGA implementation results of another recursive 2-D DCT algorithm show that the same method can be applied to implement other recursive algorithms to achieve effectively a significant hardware reduction.

5 Acknowledgement

The authors thank Dr. Bin-Da Liu and Dr. Che-Hong Chen for their valuable help, Natural Science and Engineering Research Council of Canada for the research grant supporting the project, and Canadian Microelectronics Corporation for the support of CAD and FPGA resources.

6 References

[1] Rao, K., and Yip, P.: 'Discrete Cosine Transform Algorithms' (Advantages Applications. New York: Academic, 1990)

[2] Jain, A.: 'Fundamentals of Digital Image Processing' (Prentice Hall, Englewood Cliffs, NJ: 1989)

[3] ISO/IEC JTC1/SC2/WG8, JPEG-8-R8: 'JPEG Technical Specification', 1990

[4] ISO/IEC DIS 10 918-1: 'Digital Compression and Coding of Continuous-Tone Still Images', 1992

[5] ISO/IEC JTC1/SC2/WG11, MPEG 90/176: 'Coding of Moving Pictures and Associated Audio', 1990

[6] Ruetz, P., Tong, P., Bailey, D., Luthi, P.' and Ang, P.: 'A high-performance full-motion video compression chip set', *IEEE Trans. Circuits Syst. Video Technol*., 1992, **2**, (2), pp. 111–122

[7] Matisetti, A., and Willson, A.: 'A 100 MHz 2-D 8 DCT-IDCT processor for HDTV applications', *IEEE Trans. Circuits Syst. Video Technol.,* 1995, **5**, (2), pp. 158–165

[8] Kim, K., and Koh, J.: 'An area efficient DCT architecture for MPEG-2 video encoder', *IEEE Trans. Consumer Electron*., 1999, **45**, (1), pp. 62–67

[9] Gong, D., He, Y., and Cao, Z.: 'New Cost-Effective VLSI Implementation of a 2-D Discrete Cosine Transform and Its Inverse', *IEEE Trans. Circuits Syst. Video Technol.*, 2004, **14**, (4), pp. 405-415

[10] Haque, M.: 'A two-dimensional fast cosine transforms', *IEEE Trans. Acoust., Speech, Signal Processing*, 1985, **33**, (6), pp. 1532–1539

[11] Hou, H.: 'A fast recursive algorithms for computing the discrete cosine transform', *IEEE Trans. Acoust., Speech, Signal Processing*, 1987, **35**, (10), pp. 1455–1461

[12] Yang, J., and Fan, C.: 'Fast structural two dimensional discrete cosine transform algorithms', *IEICE Trans. Fund. Electron. Commun. Comput. Sci.*, 1998, **81**, (6), pp. 1210–1215

[13] Cho, N., Yun, I., and Lee, S.: 'On the regular structure for the fast 2-D DCT algorithm', *IEEE Trans. Circuit Syst. II*, 1993, **40**, (4), pp. 259–266

[14] Cho, N., and Lee, S.: 'Fast algorithm and implementation of 2-D discrete cosine transform', *IEEE Trans. Circuits Syst.,* 1991, **38**, (3), pp. 297–305

[15] Yang, J., Bai, B., and Hsia, S.: 'An efficient two-dimensional inverse discrete cosine transform algorithm for HDTV receivers', *IEEE Trans. Circuits Syst. Video Technol.*, 1995, **5**, (1), pp. 25–30

[16] Dai, Q., Chen, X., Lin, C.: 'Fast algorithms for multidimensional DCT-to-DCT computation between a block and its associated subblocks', *IEEE Transactions on Signal Processing*, 2005, **53**, (8), pp. 3219-3225

[17] Chan, Y., Chau, L., and Siu, W.: 'Efficient implementation of discrete cosine transform using recursive filter structure', *IEEE Trans. Circuits Syst. Video Technol*., 1994, **4**, (6), pp. 550–552

[18] Yang, J., and Fan, C.: 'Compact recursive structures for discrete cosine transform', *IEEE Trans. Circuits Syst. II,* 2000, **47**, (4), pp. 314–321

[19] Yang, J., and Fan, C.: 'Recursive implementation of discrete cosine transforms: with selectable fixed coefficient filters', *IEEE Trans. Circuits Syst. II*, 1999, **46**, (2), pp. 211–216

[20] Chen, C., Liu, B., and Yang, J.: 'Direct recursive structures for computing radix-r two-dimensional DCT/IDCT/DST/IDST', *IEEE Trans. Circuits Syst. I, Reg. Papers*, 2004, **51**, (10), pp. 2017–2030

[21] An, S., and Wang, C.: 'A recursive algorithm for 2-D'. Proc. URSI International Symposium on Signals, Systems and Electronics, Montreal, Canada, July 2007, pp.335 – 338

[22] Wang, C., and Chen, C.: 'High-Throughput VLSI Architectures for the 1-D and 2-D Discrete Cosine Transforms', *IEEE Trans. Circuits Syst. Video Technol*.,1995, **5**, (1), pp. 31-40

[23] Porto, R., and Agostini, L.: 'Project space exploration on the 2-D DCT architecture of a JPEG compressor directed to FPGA implementation'. Proc. Design, Automation and Test in Europe Conference and Exhibition, Paris, France, February 2004, pp.224 – 229

List of figures

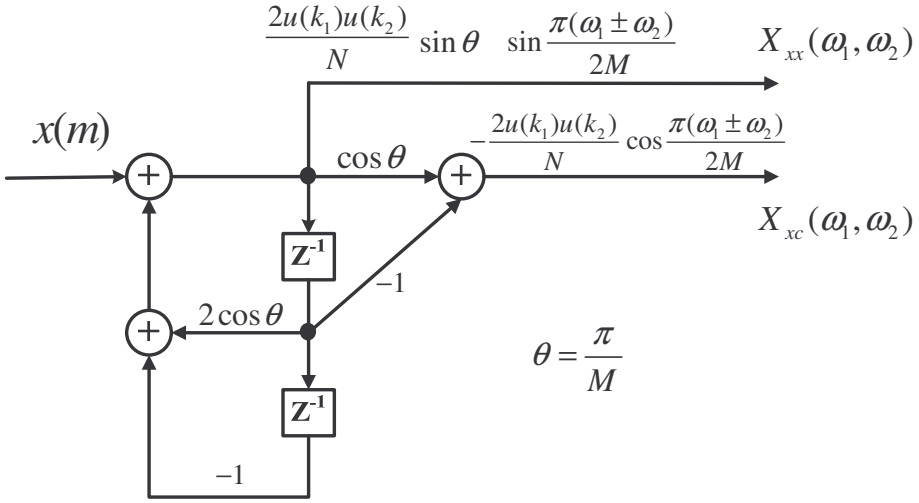

Fig.1. Recursive kernel for 1-D DCT/DST [20].

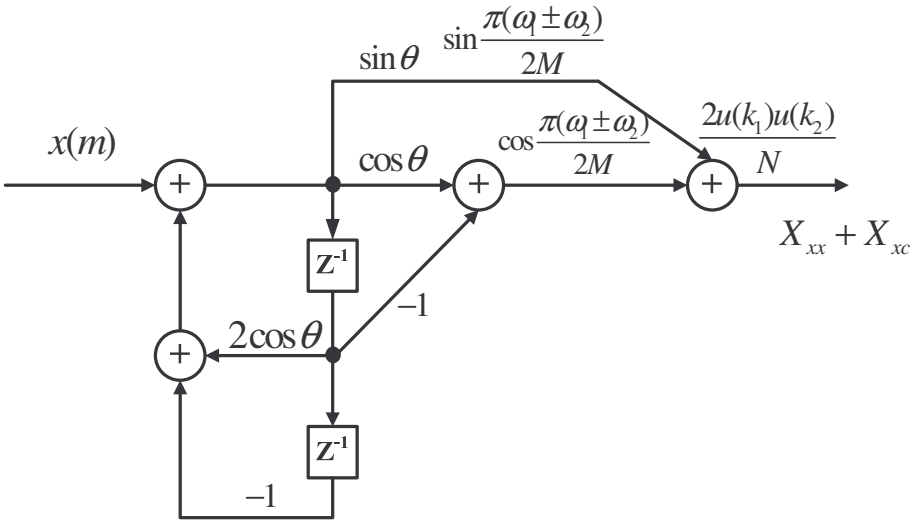

Fig. 2. Recursive kernel for the computation of the 2-D DCT defined in [20].

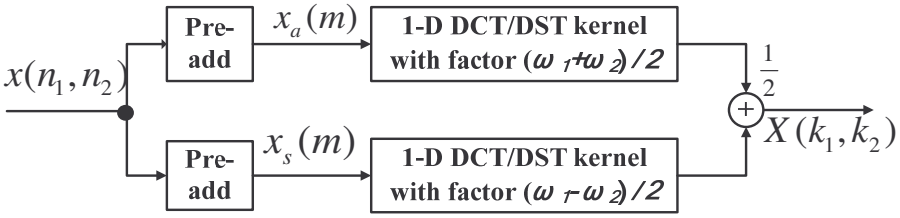

Fig. 3. Structure of the 2-D DCT computation according to the algorithm of [20]. It includes the units for the pre-addition and the recursive kernels for 1-D DCT/DST. The detail of the kernel is shown in Fig. 2.

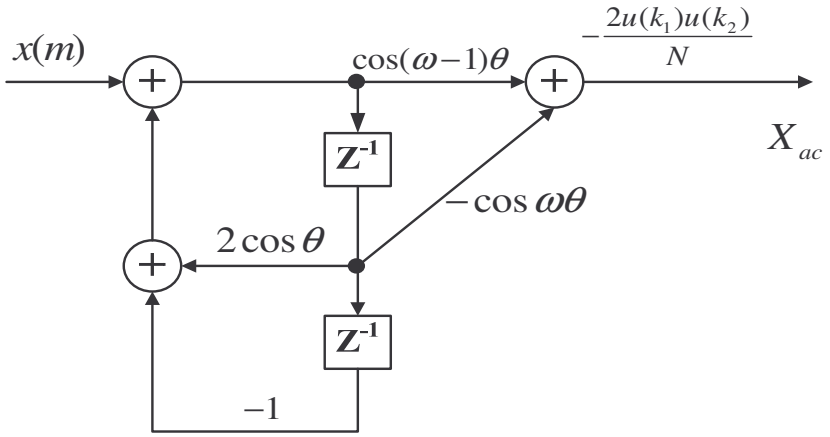

Fig. 4. Proposed recursive kernel.

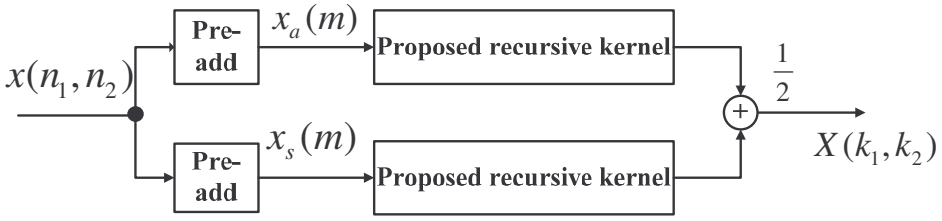

Fig. 5. Computation structure of the proposed algorithm for the 2-D DCT using the proposed recursive kernel shown in Fig. 4.

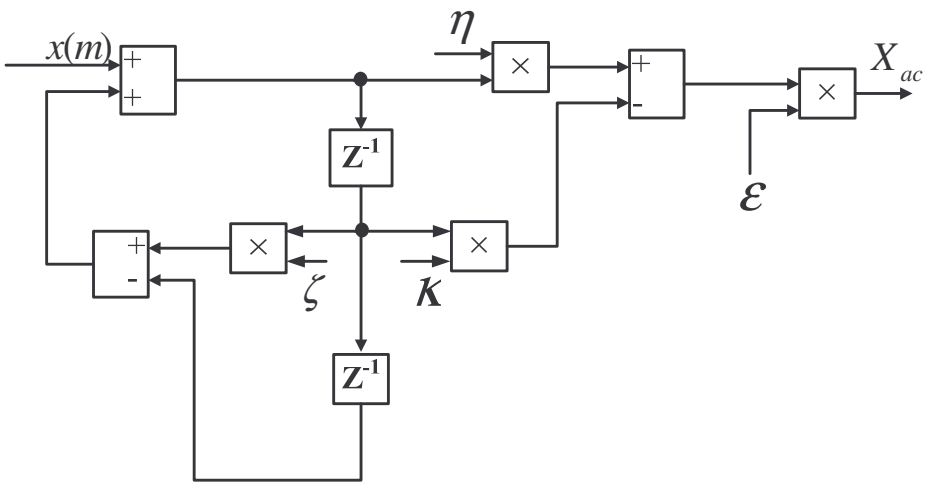

Fig. 6. Structure of the circuit block for the proposed recursive kernel. In this structure, the coefficient inputs are mapped from the kernel shown in Fig. 4, i.e. $\eta = \cos(\omega\text{-}1)\theta$, $\kappa = \cos\omega\theta$, $\varepsilon = -2u(k_1)u(k_2)/N$, and $\zeta = 2\cos\theta$.

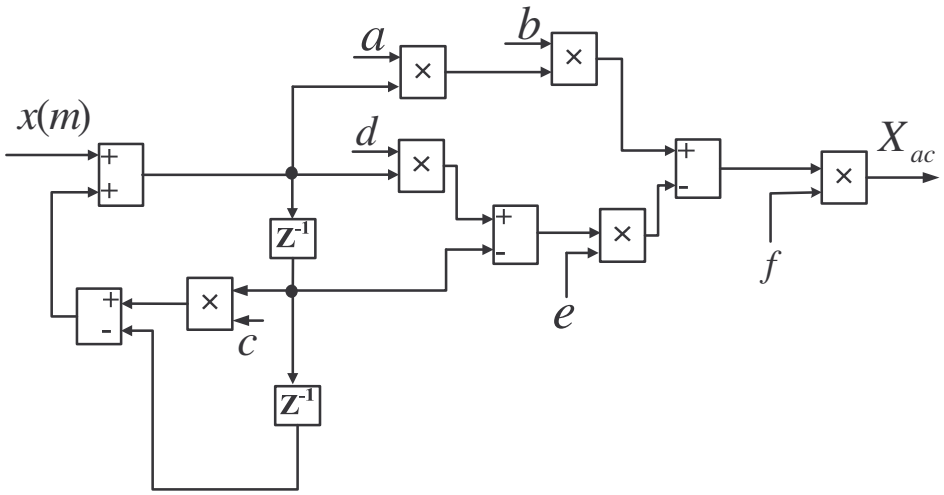

Fig. 7. Structure of the circuit block for the recursive kernel of [20]. In this structure, the inputs, a = $\sin\theta$, b = $\sin((\omega_1 \pm \omega_2)\pi/2M)$, c = $2\cos\theta$, d = $\cos\theta$, e = $\cos((\omega_1 \pm \omega_2)\pi/2M$, and f = $-2u(k_1)u(k_2)/N$.

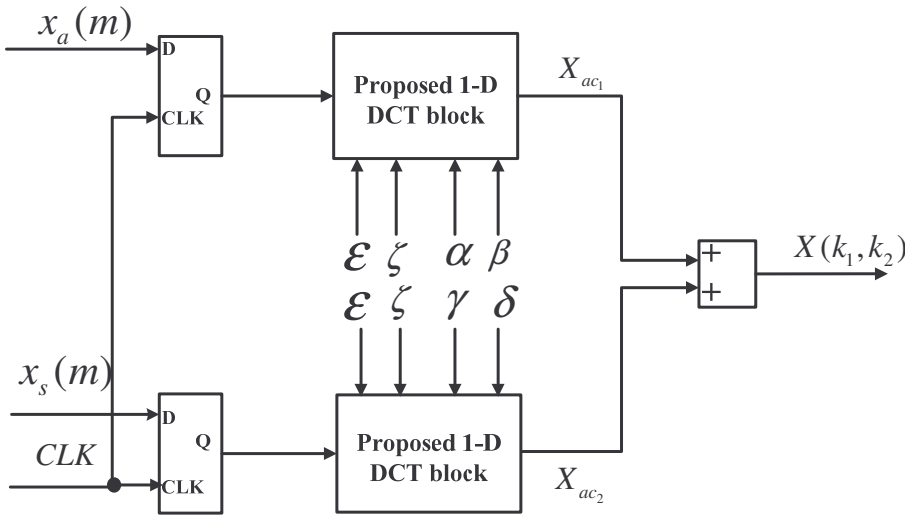

Fig. 8. Proposed architecture for the 2-D DCT computation. The structure of the proposed 1-D DCT block is shown in Fig. 6. The inputs $x_a(m)$ and $x_s(m)$ are generated by a pre-addition block.

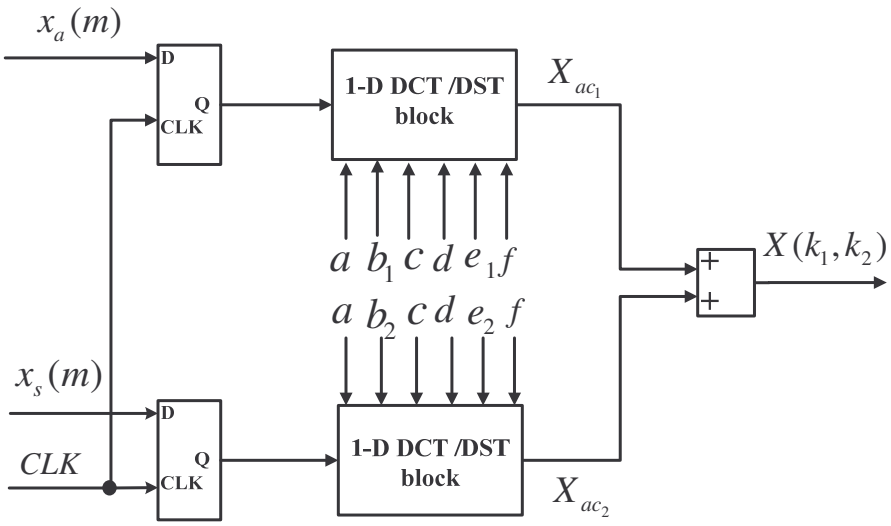

Fig. 9. Architecture for the 2-D DCT algorithm of [20]. The structure of the 1-D DCT/DST block is shown in Fig. 7.

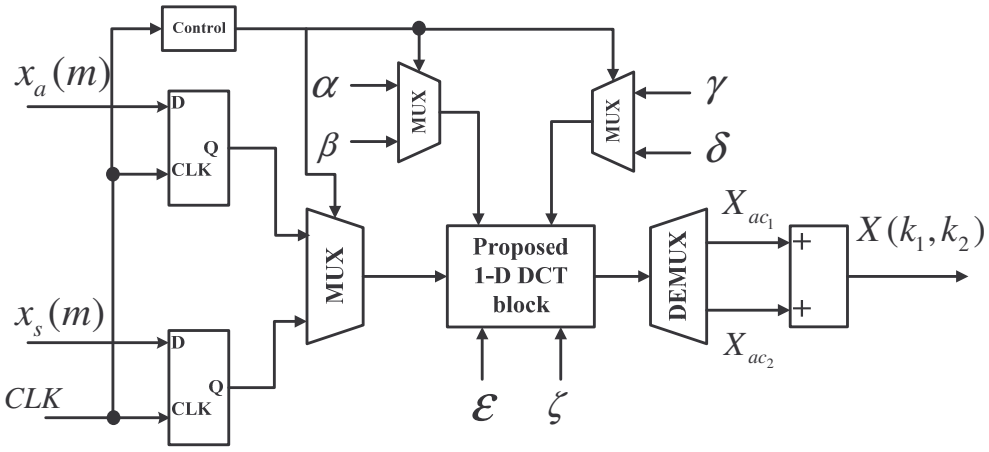

Fig. 10. Architecture for the 2-D DCT proposed for further improvement of hardware consumption. It has the single processing core, the 1-D DCT block, to compute X_{ac1} and X_{ac2}. In this structure, α=cos((ω_1+ω_2)/2-1)π/M, β=cos((ω_1-ω_2)/2-1)π/M, γ=cos((ω_1+ω_2)π/2M), δ=cos((ω_1-ω_2)π/2M), ε=-u(k_1)u(k_2)/N, and ζ=2cosπ/M.

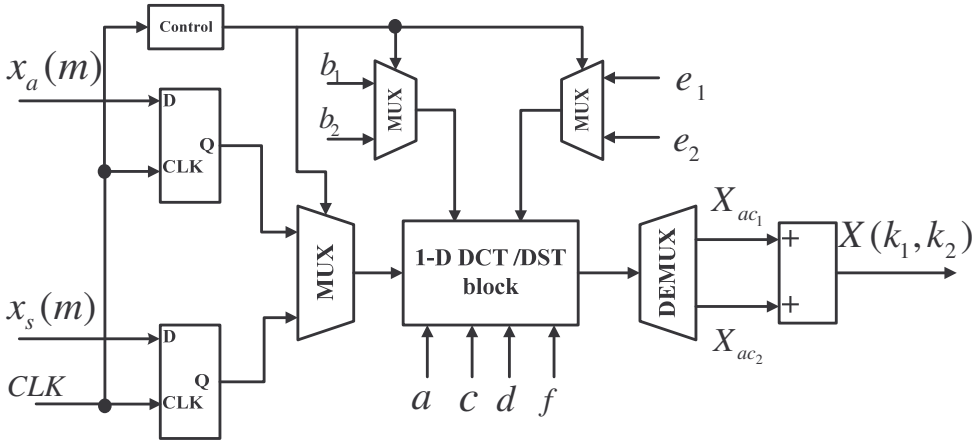

Fig. 11. Architecture for the implementation of the 2-D DCT algorithm [20]. This structure includes only one 1-D DCT/DST block as the processing core. Its pre-computed inputs are $a = \sin\theta$, $b_1 = \sin((\omega_1+\omega_2)\pi/2M)$, $b_2 = \sin((\omega_1-\omega_2)\pi/2M)$, $c = 2\cos\theta$, $d = \cos\theta$, $e_1 = \cos((\omega_1+\omega_2)\pi/2M)$. $e_2 = \cos((\omega_1-\omega_2)\pi/2M)$, and $f = -u(k_1)u(k_2)/N$.