2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Implementation of CNN-based FFT/IFFT Algorithms on FPGA

Hiep Hoang Le, Nguyen Tien Dzung, Thang Manh Hoang

Faculty of Electronics and Telecommunications
Hanoi University of Science and Technology
1 Dai Co Viet, Hanoi, VIETNAM
Email: thang@ieee.org

Abstract-- This paper describes realization of FFT/IFFT algorithms on the CNN model. The implementation for the model is carried out using kit DE2 of Altera and the experimental result demonstrates the effectiveness of proposed model.

1. Introduction

Nowadays, Discrete Fourier Transform/Inverse Discrete Fourier Transform (DFT/IDFT) is used in many applications in the information processing area such as electronics and telecommunications, digital signal processing, and spectral analysis, etc. Due to the complexity in computation of practical applications with large input, realization of DFT/IDFT requires a system with highly computational performance. It is well-known that FFT/IFFT [1] algorithms are useful methods which help to significantly reduce computation time of DFT/IDFT. In most cases, FFT/IFFT has been computed in a sequential fashion such as in hardware platform of digital signal processors, in computers. As a result, it is difficult to get a high speed FFT/IFFT since a large number of elements are used. The speed requirement on the FFT/IFFT can be seen in high speed signal processing systems such as radar, missile, optical communications, etc.

Cellular neural network (CNN) [2] consists of multiple cells which plays a role as processors running in parallel fashion. A cell in CNN locally connects with neighbor ones. CNN offers a potential of real parallel computing. So far, investigation on the idea of utilizing CNN to solve the issue of speed in FFT/IFFT has been limited [3]. As shown in [3], FFT combining with DTCNN (discrete-time CNN) has been proposed and proved theoretically. Cell neighborhood is defined in a sense of functionality, rather than topology which is possible to build many applications, thus it provides extending possibilities of CNN[3].

In this paper, implementation of 16-FFT on CNN will be described by modeling and simulation. The structure of a 16-element input array FFT using CNN model is designed on Quartus II 9.0 SP2. It is implemented on FPGA platform. Due to the similarity in FFT and IFFT algorithms, only concern to FFT is discussed.

2. CNN-Based FFT/IFFT

The basic idea was proposed by Martin Perko et al. [3] and it can be summarized as follows. The DFT formula is

$$X[k] = \sum_{n=0}^{N-1} x[n].W_N^{kn} \text{ where } W_N^{kn} = e^{-i\frac{2\pi kn}{N}} \quad (1)$$

A decimation-in-time (DIT) radix-4 FFT [1] algorithm is applied in case that N is limited to power of 4 as

$$\begin{aligned} X[k] &= \sum_{r=0}^{N/2-1} x[2r].W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1].W_{N/2}^{kr} \\ &= X_{even}[k] + W_N^k X_{odd}[k] \\ &= \left(X_A[k] + W_{N/2}^k X_B[k]\right) + W_N^k (X_C[k] W_{N/2}^k X_D[k] \\ &= X_A[k] + (W_N^k)^2 X_B[k] + W_N^k X_C[k] + (W_N^k)^3 X_D[k] \end{aligned} \quad (2)$$

Based on the above algorithm, a four-terminal operator as shown in Fig. 1 has been proposed in order to combining FFT algorithm with CNN.

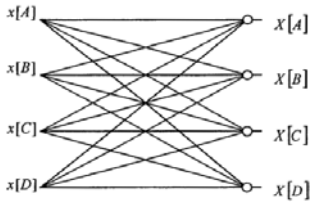

Figure 1. A four-terminal operator.[3]

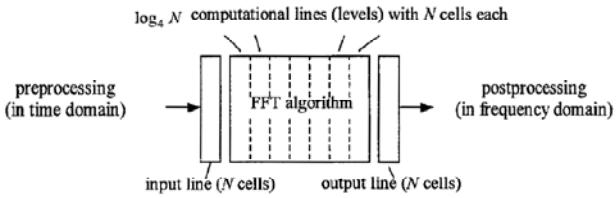

Figure 2. Illustration of CNN array for N- element FFT or IFFT [3]

Where {x[A], x[B], x[C], x[D]} are complex input arrays and {X[A],X[B],X[C],X[D]} are complex output arrays. The relations between them can be expressed as follows:

$$\begin{aligned} X[A] &= x[A] + W_{offs}^2 x[B] + W_{offs}^1 x[C] + W_{offs}^3 x[D] \\ X[B] &= x[A] - W_{offs}^2 x[B] - i.W_{offs}^1 x[C] + i.W_{offs}^3 x[D] \\ X[C] &= x[A] + W_{offs}^2 x[B] - W_{offs}^1 x[C] - W_{offs}^3 x[D] \\ X[D] &= x[A] - W_{offs}^2 x[B] + i.W_{offs}^1 x[C] - i.W_{offs}^3 x[D] \end{aligned} \quad (3)$$

Let consider an array of N-element inputs; N is limited to power of 4. Utilization of CNN array for N- element FFT or IFFT is shown in Fig. 2. $L = \log_4 N$ is the number

of computational level needed. Each level have $\frac{N}{4}$ four-terminal operators (or blocks of cells when we using CNN).

Factors W_{offs} matched with each block cell can be calculated as:

$$W_{offs}(n,l) = W_N^{n.l} \quad (4)$$

where n range from 0 to $\frac{N}{4} - 1$.and l (computation level) range from 0 to L-1. Fig. 3 shows a 16-FFT using CNN model which has 2 computation lines, 16 complex inputs and 16 complex outputs. Detail of links between output of 1st computation line and input of 2nd computation line of the model can be seen in Table 1 (the outputs and inputs are numbered as in Fig. 3). In a large model, links between two near computation lines can be determined by radix-4 Cooley-Turkey algorithm.

While a normal FFT algorithm calculates one operation at a time, it is seen that CNN-based FFT model performs $N\log_4 N$ operations simultaneously. It is easy to realize that improvement of computation performance is significant in the case that N is large.

3. Implementation of CNN-based FFT/IFFT on FPGA

We describe a 16-FFT using CNN model which is developed based on ideas as shown on previous section with hardware description language, VHDL. The chosen tool is Altera's Quartus II 9.0 SP2. Structure of the proposed model can be seen in Fig. 4. It includes 8 blocks of cells; each block contains 4 pairs of cells and each pair has 2 cells. In a pair of cells, one cell deals with real value and the other does with imaginary value of complex input. Only inputs of cells in a certain block affect states of other cells in the same block.

Cells in the same computation line are synchronized by a source of clock pulse. Characteristic equations for a cell are defined as follow:

$$\begin{aligned} x_i[k+1] &= \textstyle\sum_{j=1}^{8} B(i,j)\, u_j[k] \\ |u_j[k]| &\le 1 \\ y_i[k+1] &= f(x_i[k]) \\ f(u) &= \tfrac{1}{16}(|8+u| - |8-u|) \end{aligned} \quad (5)$$

Where $u_j[k]$ is input value of cell j (j ranges from 1 to 8) in a block cell , $x_i[k+1]$ is state of cell *i* (i also range from 1 to 8), B(i,j) is input control matrix and B (i,j) is calculated from W_{offs} as given in Eq.(6). Function f(.) is usually chosen as a nonlinear function.

Detail of a block cell designed by using ALTERA's Quartus 9.0 SP2 for FPGA implementation is shown in Fig. 5. It is clear that all of eight cells have the same structure. Inputs of a block of cells are fed inconvolution block of itself, where they are multiplied with respective input control matrix B(i,j). Sum of the results is state of the cell. Output of each cell is calculated from the state by function f(.) as reported from previous section. The cells are synchronized by clock pulses.Figure 6 illustrates the implementation on FPGA, we use SRAM on kit DE2 to store sets of input and output data.

4. Result and Discussion

We have created several testbenches to verify the proposed model. Functional simulation of the model has been created using ModelSim-Altera 6.4a. The result shows in Table 2 with the comparison with that of the same input value calculated by normal FFT function using Matlab Software. There, values are converted to complex numbers. The error isdue to limit of numerical precision representing parameters for values of inputs, states, etc. In general, it is seen that the model has worked properly as desired.

We have created the same data set as one used in simulation and written it in SRAM, after the implementing the output data set has been verified and the result shows the similarity to simulation result.

The main drawback of this model is in the usage of hardware resource in case of a large model. Generally, a model with N-element inputs requires $N\log_4 N$ cells.

About Effectiveness of the CNN-based FFT model, it carries out multiple calculation operations at the same time. Specifically, $N\log_4 N$ operations can be calculated simultaneously. However, there are two factors limiting the computation performance. Firstly, additional operations are required i.e. calculation and update for state of cells and for outputs. Secondly, there is delay in synchronizing of calculation in cells. However, synchronizing also allow a larger model works properly regardless to the scale of model.

In this realization, the clock pulse is 50 MHz, and it has taken 2 clock pulses to compute a data set, i.e. data set rate is $25x10^6$ data set per second (there are 32 inputs in one data set and we used 24 bits to present one input, so data bit rate in input of serial to parallel block can reach up to32x25=800Mb/s). The result will be very impressive if a parallel input source or serial extra-high speed bit rate is created as in optical OFDM. In case of an expansion to 1024_FFT from this model, data set rate is 10^7data set/s (in theoretical, data bit rate in STP block can reach 2048x10=20Gb/s).

Table.1 Detail of links between output of 1st computation line and input of 2nd line

Output of 1st computation line	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
Input of 2nd computation line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$$B_{ij} = \begin{bmatrix} 1 & 0 & \mathrm{Re}(W_{offs}^{2}) & -\mathrm{Im}(W_{offs}^{2}) & \mathrm{Re}(W_{offs}^{1}) & -\mathrm{Im}(W_{offs}^{1}) & \mathrm{Re}(W_{offs}^{3}) & -\mathrm{Im}(W_{offs}^{3}) \\ 0 & 1 & \mathrm{Re}(W_{offs}^{2}) & \mathrm{Im}(W_{offs}^{2}) & \mathrm{Re}(W_{offs}^{1}) & \mathrm{Im}(W_{offs}^{1}) & \mathrm{Re}(W_{offs}^{3}) & \mathrm{Im}(W_{offs}^{3}) \\ 1 & 0 & -\mathrm{Re}(W_{offs}^{2}) & \mathrm{Im}(W_{offs}^{2}) & \mathrm{Im}(W_{offs}^{1}) & \mathrm{Re}(W_{offs}^{1}) & -\mathrm{Im}(W_{offs}^{3}) & -\mathrm{Re}(W_{offs}^{3}) \\ 0 & 1 & -\mathrm{Im}(W_{offs}^{2}) & -\mathrm{Re}(W_{offs}^{2}) & -\mathrm{Re}(W_{offs}^{1}) & \mathrm{Im}(W_{offs}^{1}) & \mathrm{Re}(W_{offs}^{3}) & -\mathrm{Im}(W_{offs}^{3}) \\ 1 & 0 & \mathrm{Re}(W_{offs}^{2}) & -\mathrm{Im}(W_{offs}^{2}) & -\mathrm{Re}(W_{offs}^{1}) & \mathrm{Im}(W_{offs}^{1}) & -\mathrm{Re}(W_{offs}^{3}) & \mathrm{Im}(W_{offs}^{3}) \\ 0 & 1 & \mathrm{Re}(W_{offs}^{2}) & \mathrm{Im}(W_{offs}^{2}) & -\mathrm{Re}(W_{offs}^{1}) & -\mathrm{Im}(W_{offs}^{1}) & -\mathrm{Re}(W_{offs}^{3}) & -\mathrm{Im}(W_{offs}^{3}) \\ 1 & 0 & -\mathrm{Re}(W_{offs}^{2}) & \mathrm{Im}(W_{offs}^{2}) & -\mathrm{Im}(W_{offs}^{1}) & -\mathrm{Re}(W_{offs}^{1}) & \mathrm{Im}(W_{offs}^{3}) & \mathrm{Re}(W_{offs}^{3}) \\ 0 & 1 & -\mathrm{Im}(W_{offs}^{2}) & -\mathrm{Re}(W_{offs}^{2}) & \mathrm{Re}(W_{offs}^{1}) & -\mathrm{Im}(W_{offs}^{1}) & -\mathrm{Re}(W_{offs}^{3}) & \mathrm{Im}(W_{offs}^{3}) \end{bmatrix} \quad (6)$$

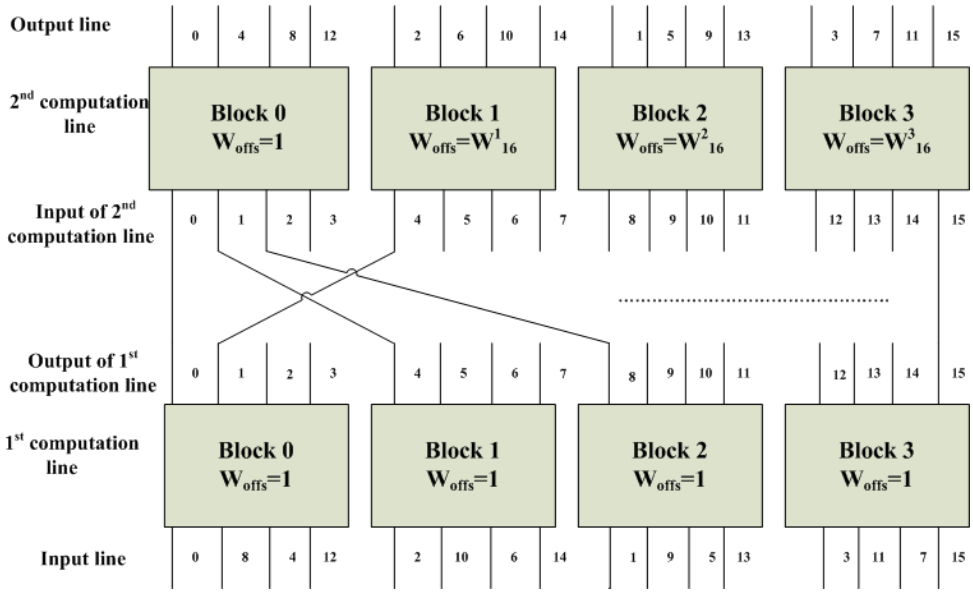

Figure 3. 16-FFT using CNN model

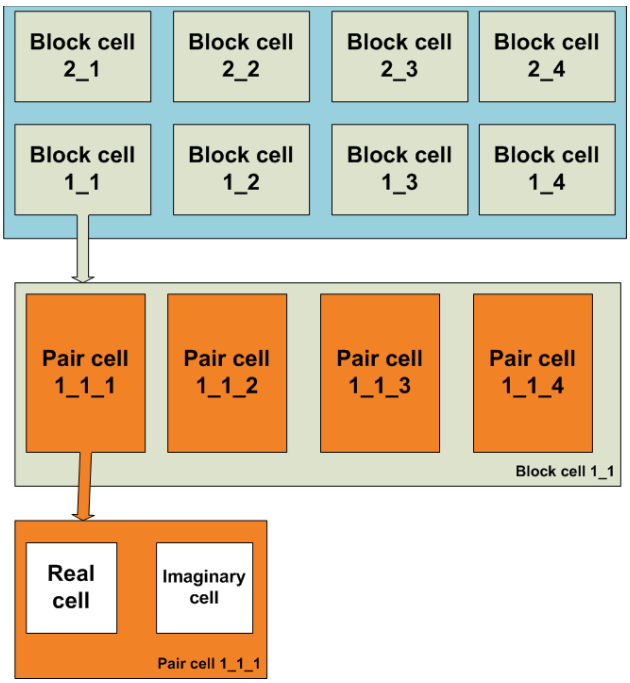

Figure 4. Block diagra describes structure of the model

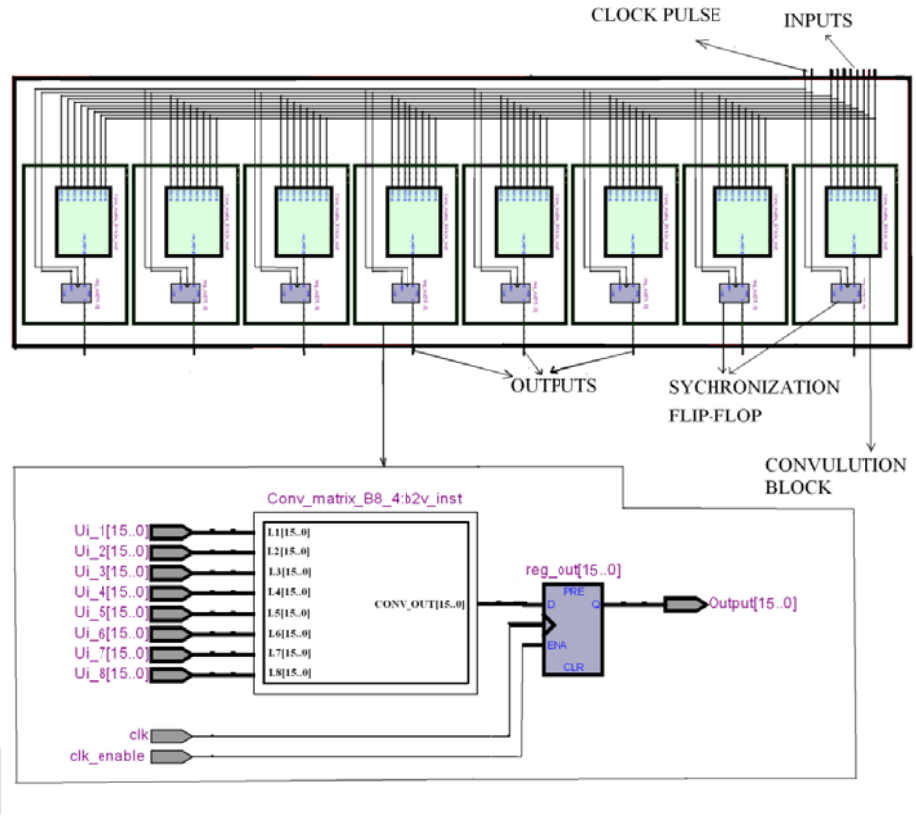

Figure 5. Detail of a block cell designed by using ALTERA'S Quartus 9.0 SP2 for FPGA.

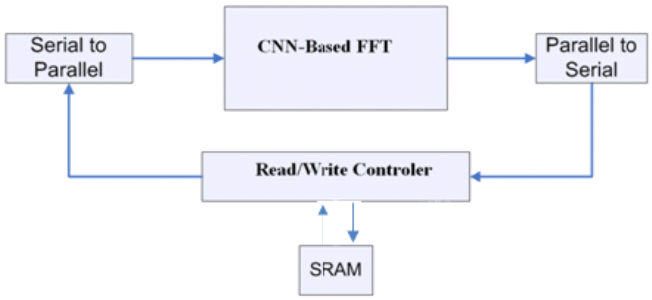

Figure 6 .Block diagram describes the implementation on FPGA.

5. Conclusion

The paper presents implementation of CNN-based FFT/IFFT on FPGA. The result has shown that implementation of the model gives a potential extra-high speed computation. Furthermore, the proposed model can be developed to larger number of N for various applications.

Table 2. Simulation result.

Input	Output (using Matlab)	Output (using ModelSim)
0.9293	8.7960	8.79586
0.3500	-0.4787 + 0.9380i	-0.4787 +0.9379i
0.1966	0.5192 +0.4591i	0.51927+ 0.4590i
0.2511	0.4291 +0.2097i	0.42917+ 0.20955i
0.6160	1.0419 -0.1912i	1.0418 -0.19119i
0.4733	1.2373 -0.5678i	1.2373 -0.5677i
0.3517	-0.2366 +1.2225i	-0.23648+ 1.2224i
0.8308	0.1883 -0.4041i	0.18819 -0.40415i
0.5853	0.6714	0.6715
0.5497	0.1883+0.4041i	0.18819+ 0.40413i
0.9172	-0.2366-1.2225i	-0.23648 -1.2224i
0.2858	1.2373 +0.5678i	1.2373 +0.56768i
0.7572	1.4019+0.1912i	1.04189 +0.19119i
0.7537	0.4291 -0.2097i	0.42917 -0.20957i
0.3804	0.5192 -0.4591i	0.51927 -0.45904i
0.5678	-0.4787-0.9380i	-0.47871 -0.93793i

Acknowledgments

This work has been supported by Ministry of Science and Technology of Vietnam with the project number ĐTĐL2009G/44. The authors would like to thank Vietnam's National Foundation for Science and Technology Development (NAFOSTED).

References

[1] C.S. Burrus and T.W. Parks, "DFT/FFT and Convolution Algorithms." New York: John Wiley & Sons, 1985.

[2] Leon O.Chua and Lin Yang "Cellular Neural Networks:Theory", IEEE Trans. CAS,vol. 35, pp. 1257-1272, 1988.

[3] Martin Perko, Iztok Fajfar, Tadej Tuna, and Janez Puhan, " Fast Fourier Transform Computation Using a Digital CNN Simulator ", the Fifth IEEE International workshop on Cellular Neural Networks and their Applications, London. England, 14-17 April 1998, pp. 230-235.