Parallel FFT Computation with a CDMA-Based Network-on-Chip

Daewook Kim, Manho Kim and Gerald E. Sobelman
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA
{daewook,mhkim,sobelman}@ece.umn.edu

***Abstract*— Fast Fourier transform (FFT) algorithms are used in a wide variety of digital signal processing applications and many of these require high-performance parallel implementations. We present two methodologies for mapping an FFT computation onto a CDMA-based star topology network-on-chip (NoC) architecture. These implementations reduce the FFT data shuffling time and simplify the data flow between processing elements. The design has been modeled using SystemC and the simulation results provide throughput and latency performance metrics for the different mapping scenarios.**

I. Introduction

Many prior approaches for parallel FFT implementation have been proposed, such as a two-dimensional (2D) mesh array [1] and the perfect shuffle network [2]. While these designs have various advantages, they also have limitations when scaled up to large-size transforms. For example, the size of the 2D mesh array increases in proportion to size of the transform [1]–[3]. Although large FFT calculations can be folded onto smaller-size meshes, this is achieved at a cost of complex, overlapped communication between processing elements (PEs). Moreover, the inter-PE communication combined with the data loading/unloading time can become a bottleneck and may dominate the execution time.

In this paper, we consider the application of a network-on-chip (NoC) to the problem of parallel FFT computations. In other words, we consider a multi-processor system-on-chip (SoC) in which the PEs are interconnected using an NoC to obtain a high-throughput parallel FFT implementation. Several different types of NoC architectures [4]–[9] have been proposed to overcome the bandwidth limitations of bus based interconnections, and each one has a particular set of performance limitations and overhead characteristics. For example, the number of switches required in a mesh structure increases with an increasing number of PEs, which leads to greater area overhead and power consumption. In addition, complex routing algorithms through the network of switches may be necessary to achieve good performance, which also has an adverse impact on area and power.

The architecture proposed in this paper originated from the desire to address these issues. In this study, we utilize an NoC architecture that is based on code division multiple access (CDMA) techniques [10]. CDMA has been widely used in wireless communications networks because of its bandwidth capacity and multi-user features, but it has only rarely been applied in the context of a wired network. The CDMA-based NoC architecture utilizes the orthogonality property of Walsh codes to route data packets from source to destination. An attractive aspect of the CDMA-based NoC is that the inter-PE communication is accomplished using a small number of switches, in some cases just a single switch. In the remainder of this paper, we will show how FFT algorithms can be mapped onto this NoC architecture and give simulation results for throughput, latency and computational response time.

II. Network-on-Chip Architecture

A CDMA-based star network topology which can accommodate up to 8 PEs is shown in Fig. 1. If a larger number of PEs are needed then a hierarchical star network topology can be formed consisting of a central switch connected to a set of local switches, where a set of PEs are connected to each local switch [10]. The number of PEs attached to a switch dictates the required length of the Walsh codewords that are used. Two important design simplifications can be applied for the case of FFT computations.

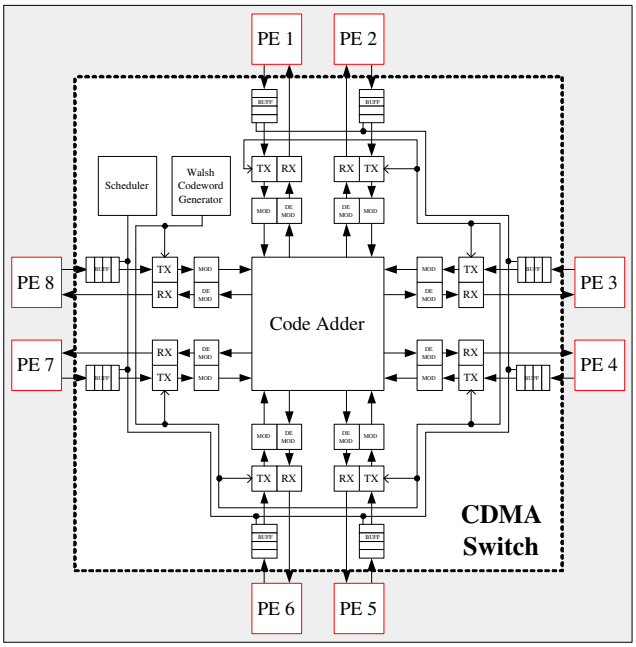

Fig. 1. CDMA-based star topology NoC architecture.

First, there will not be any packet address contention issues because the result of each PE is sent to a different destination PE according to the FFT data-flow diagram. Second, we don't have to consider the case in which a PE has no data to send since the regularity of the FFT computation ensures that each PE will be continuously utilized. Packets are composed of three fields, consisting of the source address, the destination address and the payload. Each data value is loaded into the payload field as a serial bit stream.

For concreteness, we consider the specific example of a 16-point FFT. Each of the 8 PEs of Fig. 1 is capable of computing 4 butterflies, either in parallel or in sequence, depending on the mapping methodology that is used. The spreading code used in our design is an 8-chip orthogonal Walsh code which provides sufficient codeword orthogonality to allow the 8 PEs to simultaneously communicate with each other.

A. Transmitter (TX) and Modulator (MOD)

The TX block receives a packet from a buffer and examines its destination address. This information is used to choose the appropriate Walsh codeword that corresponds to the required destination. If L-bit codewords are used, we can assign each of the L-1 non-zero codewords to each of L-1 PEs, with the all-zero codeword reserved for the "no data sent" case. The corresponding modulation rule is given in Table I. On the other hand, when we have an application such as the FFT in which that case does not arise, then the all-zero codeword can also be assigned to one of the PEs so that an L-bit codeword can support up to L PEs.

TABLE I
MODULATION ALGORITHM

Data	*Codeword Assignment*
0	Codeword itself
1	Inverted codeword
No data	All-zero codeword

B. Demodulator (DEMOD) and Receiver (RX)

The demodulator recovers the original data by computing a summation and then applying a decision rule. The demodulation algorithm uses a decision factor called λ, which is a modification of a previously proposed scheme [11] that is designed to work with Walsh codewords.

The mathematical equations and algorithms necessary for demodulation are summarized in the equations below together with the decision rule of Table II.

$$D[i] = \begin{cases} (2S[i] - L) & \text{if codeword}[i] \text{ is 0} \\ (-2S[i] + L) & \text{if codeword}[i] \text{ is 1} \end{cases} \quad (1)$$

$$\lambda = \sum_{i=0}^{L-1} \frac{D[i]}{L} \quad (2)$$

- S[i] is the summation of all modulated values.
- L is the codeword length.
- D[i] is the decision variable.
- λ is the decision factor.
- $0 \leq i \leq L-1 (i : integer)$

TABLE II
DEMODULATION ALGORITHM

Decison Factor(λ)	*Demodulated Data[bit]*
+1	1
-1	0
0	No data sent

Table III gives a numerical example of the modulation and demodulation process. Suppose that PE 5 wishes to send a data value 0 to PE 3. The assigned codeword [0 0 1 1 0 0 1 1] itself is used because the transmitted data is 0 in this case. The other PEs also send 0 or 1 simultaneously in a similar manner. In this example, the S[i] values are [3 3 3 7 5 5 5 5]. The only information that the demodulator associated with a PE knows are S[i] and its assigned codeword. Each digit of S[i] is doubled, which results in [6 6 6 14 10 10 10 10]. The decision variable D[i] outputs values that correspond to each digit of original codeword, which results in [-2 -2 2 -6 2 2 -2 -2]. Therefore, we can see that the decision factor λ results in -1 for destination 3. This indicates that the recovered data bit is 0, which is precisely the value that was sent from source 5.

TABLE III
DEMODULATION EXAMPLE

src no	src data	dst no	mod codeword	original codeword	λ	rcv data	dst no
1	1	8	10010110	00000000	+1	1	1
2	0	5	00001111	01010101	-1	0	2
3	0	6	01011010	00110011	-1	0	3
4	0	2	01010101	01100110	+1	1	4
5	0	3	00110011	00001111	-1	0	5
6	1	1	11111111	01011010	-1	0	6
7	1	4	10011001	00111100	-1	0	7
8	0	7	00111100	01101001	+1	1	8
Summation(S[i])=[3 3 3 7 5 5 5 5]							

III. MAPPING THE FFT ONTO THE NOC

We consider the case of mapping a 16-point, radix-2, decimation-in-frequency (DIF) FFT algorithm onto the proposed CDMA NoC architecture. The entire computation can be divided into three parts. First, a preprocessing step is responsible for loading the input data. Then, the FFT computation is performed by the network of PEs. All PEs accept new input data in a parallel and pipelined fashion and all data transmission between source and destination PEs are executed concurrently through the CDMA switch. Finally, postprocessing is responsible for storing the computation results. Each value is represented as a signed 16-bit fixed point number with 10 fractional bits. Once an FFT computation has been performed, the block writes the transform values to the destination PEs.

In the following two subsections, we propose and compare two different FFT mapping methodologies for use with this CDMA star topology NoC.

A. Direct Mapping

An N-point FFT requires $\log_2 N$ stages with N/2 butterfly operations at each stage. Fig. 2(a) shows the direct mapping technique for a 16-point DIF FFT onto 8 PEs, where each PE is capable of computing 4 butterflies. While the computations on processors PE3 through PE8 are done independently, the computations in PE1 and PE2 share the same input data. The inter-PEs data flow of the direct mapping technique is PE1→PE3→PE5→PE7 and PE2→PE4→PE6→PE8. The loading of input data and the inter-PEs data communication are executed in a pipelined manner for maximum throughput.

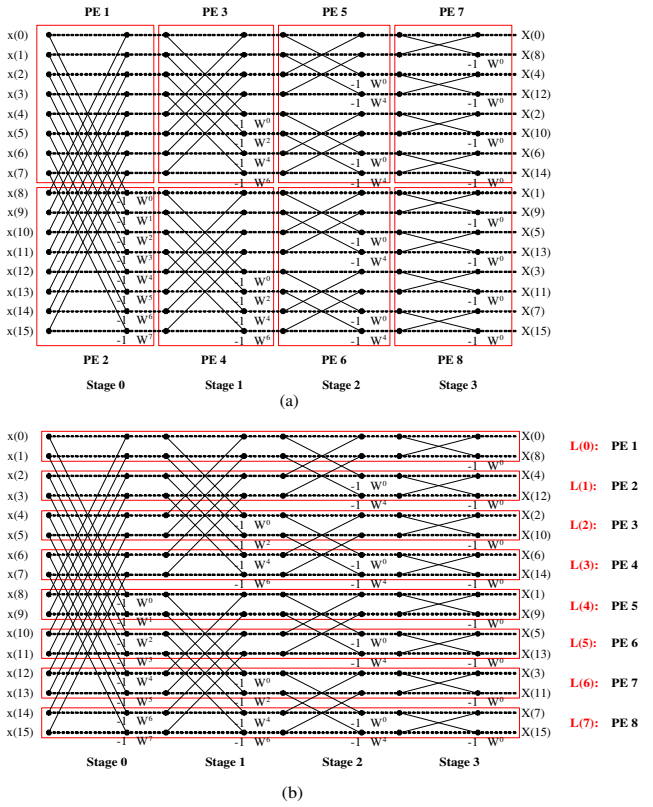

Fig. 2. (a) Direct mapping and (b) Indirect mapping.

B. Indirect Mapping

The proposed indirect mapping is to divide the overall computation task according to a set of 8 virtual data paths as shown in Fig. 2(b). One pair of input paths extends horizontally along the data-flow graph and comprises one virtual level. In this scheme, each PE iteratively computes a butterfly four times along each of the 8 virtual data paths. This type of partition has an area advantage as the size of the FFT increases, as shown in Table IV.

Note that the number of PEs used in a mesh array is the same as the number of points N in the FFT, while the indirect mapping technique requires only half of this number. The number of stages increases by one when the number of FFT points is doubled. The details of the indirect mapping algorithm are best described using an example. Consider the first level L(0) assigned to PE1 in Fig. 2(b). As shown in Fig. 3, we trace the dataflow until PE1 obtains its final computation result. In stage I, PE1 computes the initial input data pair in0 and in1 with input data in8 and in9 from PE5. Since we already know the previously assigned values of variables a, b, c, and d from the FFT algorithm, the PE1 result is updated iteratively for the input value of the next stage, as shown in Fig. 3. Continuing in this manner, PE1 obtains its final computation values for x(0) and x(8) during stage IV.

TABLE IV
NUMBER OF PES COMPARISON

FFT Point (N)	*Mesh Array [1]*	*Direct Mapping*	*Indirect Mapping*
16	16	8	8
32	32	20	16
64	64	48	32
128	128	112	64

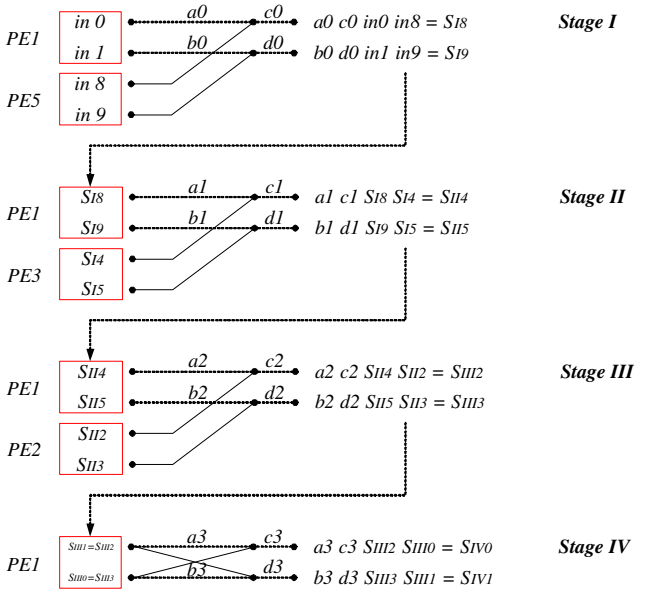

Fig. 3. Dataflow of indirect mapping.

IV. SIMULATION RESULTS AND ANALYSIS

We have simulated and analyzed the network performance of the direct and indirect FFT mapping methodologies using SystemC. We assume that the on-chip communication is synchronous and runs at the same speed as a PE, and also that the computational load has been equally distributed. The performance metrics that we have analyzed are latency, maximum throughput, average throughput, network utilization and computation response time, which are inter-related as follows:

$$Latency = Packet\ trasmission\ delay \tag{3}$$

$$Max.throughput = \frac{data\ transferred\ between\ nodes}{Latency} \times I_{max} \tag{4}$$

$$Avg.throughput = \frac{data\ transferred\ between\ nodes}{Latency} \times I_{avg} \quad (5)$$

$$Network\ utilization = \frac{Average\ throughput}{Maximum\ throughput} \quad (6)$$

$$Response\ time = Elapsed\ time\ of\ one\ FFT\ computation \quad (7)$$

where I_{max} and I_{avg} are the maximum and average number of simultaneous data transfers, respectively. Note that throughput may also be interpreted as the aggregate bandwidth of the network.

In our simulations, we set the switch system clock period to L times the codeword clock, i.e. $T_{sysclk} = L * T_{codeclk}$ and the switch operates at 64 MHz. The demodulator outputs the recovered data after some latency depending on the size of the packet payload. Fig. 4 shows the simulation results for these four performance metrics as a function of the packet payload size. From part (a) we can see that the latency increases for increased packet sizes, as expected, and that both the direct and indirect mapping methods give the same values for this metric. Parts (b), (c) and (d) show that throughput and response time increase with increasing packet size. Moreover, throughput is higher with the indirect mapping method at the expense of increased response time. This longer response time is due to the data dependencies that exist in the indirect mapping approach, as described earlier.

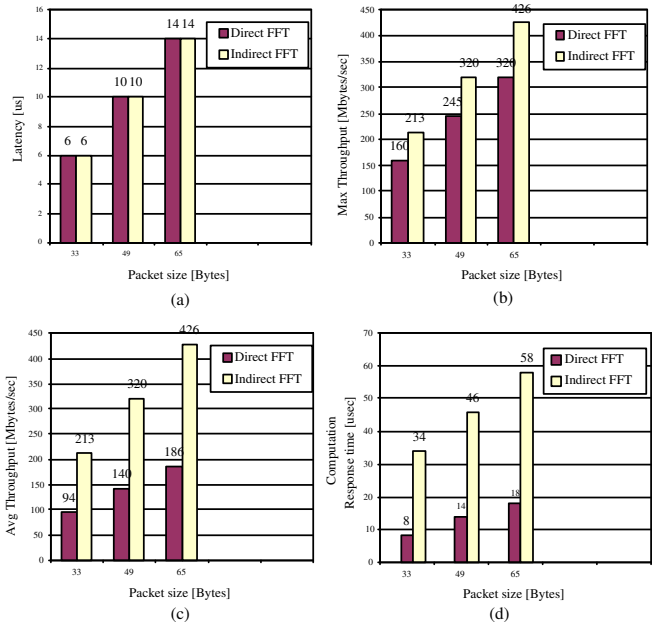

Fig. 4. Simulation results: (a) Latency[μsec], (b) Max. throughput[Mbytes/sec], (c) Avg. throughput[Mbytes/sec], and (d) Computation response time[μsec].

V. Conclusions

In this paper, the mapping of a parallel FFT algorithm onto a CDMA-based star topology network-on-chip architecture has been presented. Walsh codes are used to modulate the packet data and the data communications between processing elements occurs concurrently without interference. Two mapping methodologies have been devised for allocating the PEs to portions of the FFT data-flow diagram. Simulations have been performed using SystemC and the results show the relationships between various performance metrics. In our future work, we plan to model and synthesize an MPEG-4 system based on our CDMA-based NoC platform. We also plan to investigate the use of sophisticated scheduling algorithms to ensure fairness and to achieve guaranteed throughput.

Acknowledgements

We thank Sangwoo Rhim, Bumhak Lee and Euiseok Kim of the Samsung Advanced Institute of Technology (SAIT) for their help with this manuscript. This research work is supported by a grant from SAIT.

References

[1] M. Lee, K. Shin, and J. Lee, "A VLSI array processor for 16-point FFT," *IEEE Journal of Solid-State Circuits*, Vol. 3, Sept 1991, pp. 1286–1292.

[2] H. Stone, "Parallel processing with the perfect shuffle network," *IEEE Trnas. Comput.*, Vol. C-20, Dec 1986, pp. 153–161.

[3] J. W. Jang and W. Przytula, "Trade-offs in mapping fft computations onto fixed size mesh processor array," *5th International Parallel Processing Symposium. Proceedings*, 1991.

[4] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," *Design, Automation and Test in Europe Conference and Exhibition. Proceedings*, 2002.

[5] S. Kumar, A. Jatsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," *Proc. of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2002.

[6] J. Soininen, A. Jantsch, M. Forsell, A. Pelkonen, J. Kreku, and S. Kumar, "Extending platform-based design to network on chip systems," *Proc. of 16th International Conference on VLSI Design*, 2003.

[7] P. P. Pande, C. Grecu, A. Ivanov, and R. Saleh, "Design of a switch for network on chip applications," *Proc. of the 2003 International Symposium on Circuits and Systems*, Vol. 5, May 2003, pp. v217–v220.

[8] J. Liu, L. R. Zheng, and H. Tenhunen, "A guaranteed-throughput switch for network-on-chip," *Proc. of the System-on-Chip International Symposium*, pp. 31–34, Nov 2003.

[9] C. A. Zeherino, and A. A. Susin, "SoCIN: a parametric and scalable network-on-chip," *Proc. of the 16th Integrated Circuits and Systems Design Symposium*, pp 169–174, Sep 2003.

[10] D. Kim, M. Kim and G. E. Sobelman, "CDMA-based network-on-chip architecture," *Proc. of the IEEE Asia Pacific Conference on Circuits and Systems*, pp. 137–140, 2004.

[11] R. H. Bell, C. Y. Kang, L. John, and J. Swartzlander, E. E., "CDMA as a multiprocessor interconnect strategy," *Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*, Vol. 2, pp. 1246–1250, Nov 2001.