Implementation of Multi-Dimensional FFTs using FPGA

G Ravi Kumar Reddy
CVR College of Engineering College, ECE Department, Hyderabad, India.
Email: ravigrkr@gmail.com

Abstract—**This paper presents the implementation of multidimensional Fast Fourier Transform (FFT). In this implementation the FFT is calculated separately per each dimension in the pipeline. The resultant three dimensional pipeline FFT is implemented on an FPGA. The implemented FPGA can calculate the 3-dimensional (3D) FFT for a data which consists of 2563 samples with each sample with word size of 32- bits. The main challenge of this paper is the permutations between the one dimensional FFT modules. For these permutations, the storage memory is an external DDR2 SDRAM and on chip memory BRAM. The resultant multi-dimensional FFT is hardware efficient with very less latency around 84.2 msec.**

Index Terms—**DFT, FFT, FPGA, RAM, SDRAM & BRAM.**

I. INTRODUCTION

In the real world applications, the multi-dimensional FFTs are widely used. For example, the 2D FFT used in synthetic Aperture Radar [8], and 3D FFT used in astrophysics, motion detection cosmology, turbulence simulations[7], molecular dynamics and reverse tomography. At present, the iterative 2D and 3D FFTs are only implemented in the hardware, but the pipelined architectures are placed in 2D FFTs only. The aim of this paper is to design and implement a pipeline 3D FFT with stable throughput using external memory for permutations by supporting large number of input data sets. The motivation for this is to enable the use of 3D FFT in real-time systems demanding constant throughput of samples.

Generally, the discrete Fourier transform (DFT)is a different type of Fourier transform in the discrete manner set. The Cooley and Tukey invented the FFT in the year of 1965 [2]. The FFT algorithm reduces the total no. of calculations required to perform DFT. Since the FFT has been mostly used and is still used and research is being carried out today also. The entire FFT implementations are two groups of categories, one is iterative and the second is pipelined architectures. In the iterative architecture, one or more processing elements are reused for the calculation of the result. In the pipelined architecture, a series of processing elements are used to calculate result from sequence of samples. The iterative method is hardware efficient because it uses same hardware for many calculations but it is not suitable for continuous flow. For continuous flow environment, the less latency is required, which is provided by pipeline architecture with more hardware [3].

The same classification strategy is followed for the multi dimensional FFT(MD FFT) architectures. The MDFFT is also grouped into iterative and pipelined architectures. The iterative architectures which are discussed in[7,9] are designed as more flexible than the available processing elements. The performance of this is same as CDFFT only. The counter are pipelined architecture which supports the continuous flow of sample calculation by using dedicated hardware like 1DFFT.

II. ARCHITECTURE SELECTION

The architecture selection completely depends on the type of application in the real time environment. The tough constraints are considered for both latency and throughput. The typical example is a processing chain, in this each and every clock cycle provides data and gives the results in the same speed with required latency.

Another example in real time applications, is the post processing of the video stream is considered, in which the decision marker cannot take the correct decision in time if the latency is high. One more example is medical body scanning; for this application the doctor is needed to provide the information in time with respect to updates of the image, then low latency and higher throughputs are required.

A. Pipelined or Iterative Architecture

In real time applications there is requirement for fast computation of both 2D and 3D FFT. When we consider inside of the continuous flow, the iterative architectures halt a bit flow .The main reason for this is the memory loading and storing intermediate results by the repeated access.

For example, the 3D FFT have to read and write the data set 3 or 4times,than to process the chain and include a mandatory clock domain crossing (CDC) the bandwidth of memory also required 3 or 4 times. From the experience of CDCs, the debugging is tedious if the design is that in good manner. Hence, we can understand that the iterative process architectures are not fit for real time applications at the high cost of very advanced circuitry and large memories.

The pipelined architectures are basically adapted to a continuous flow of samples. This architecture enables the 1D FFTs to be evaluated without repeated memory accesses but hardware cost is high for mandatory permutations. The bandwidth of memory remains same for

DOI: 10.32377/cvrjst1005

the entire processing chain because memory uses only for read and write operations at once . For small data sets, the permutations with respect to on-chip memory only taken place with less hardware. The large data set permutations required external memory with some constraints and accessing limitations.

B. Memory permutations with some access limitations

In the implementation of 3- dimensional FFT, a two dimensional FFT is also considered as a part. Suppose if we are calculating the 2D FFT of each slice is spanned into two of 3 dimensions, followed by the calculation of 3D FFT. Therefore, the initial part of the 3 dimensional pipeline FFT is same as the pipelined 2D FFT as in the paper [3], namely a1D FFT followed by a transposition unit of the another 1D FFT. The transposition of permutations are present in both 2D and 3D FFTs. The 3 dimensional FFT performs another permutation between the 2nd and 3rd 1D FFT blocks, permutation has to permute the whole 3D dataset such that data is delivered in the 3 dimension to the 3rd1D FFT.

If the data is larger in size, one or both of those permutations have to perform in external memory. Hence, the large and high speed external SDRAM architecture is chosen to provide the required. The SDRAMs are dynamic in nature, so refreshing is required frequently, and you can access limited memory only at a time. This is the one difficulty while performing real time permutations for pipelined 3D and 2D FFTs. Basically, SDRAMs are burst oriented and at a time series of samples access is possible. That means the lowest address bits and corresponding elements are locked within the burst and permutated using separate auxiliary circuit. These permutations are not performed on SDRAM. In the auxiliary permutation circuit, the places of the bits are swapped and finally placed on their site.

Next, the memory SDRAM gets refreshed in a particular period of time without any loss of the data. This is considered as important constraint on the access of the memory schedule. The memory schedule can be done in two ways for refreshing, one is dynamic schedule and other one is static schedule. If dynamic schedule is selected, the latency is varied and sometimes throughput also varied, which is not supported by some applications. The other one is static schedule refreshing, in which refresh requires long time to fit with overhead by iterations with considerable bandwidth.

After that, for the rows and columns the less amount of memory access is only possible. Hence, the data is mapped to access the number of elements in all possible dimensions inside rows. For this sufficient row switches are required to overcome the overhead of the rows memory. The overhead concept is related to the process of pre-charging the active rows, the storing of the rows in the memory array, and fetching new rows and finally placing of those new rows in the sense amplifier blocks.

III. PROPOSED APPROACH

In this section, we focus on the 3D permutation between 2nd and 3rd 1D FFT. The series of 1D FFT architecture is selected with permutations in between; at the end we also perform a bit reversal result with natural order. The complete architectural view with bit reversal is shown in figure1.

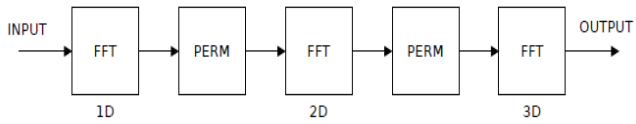

Figure 1. The complete view of the architecture for 3D FFT

The 3D permutation design process is also called as 3D rotation because this permutation rotates in three dimensions. This entire process is divided into following steps. 1) The determination of the external memory requirement, 2) identification of memory to fulfill the bandwidth requirements of the application and extraction of parameters to overcome the access 3) if external memory is not required, directly the design can perform permutations on internal memory i.e BRAM. 4) next step is determination of static schedule to avoid the tedious work to fulfill the latency and throughput always while using dynamic schedule, which includes group of memory commands for accessing.

In the design of the schedule, the row changes are not allowed during the schedule processing. If row changes are allowed, the data permutations inside the group is not performed without extra memory. Inside the group, the data index bits are locked when memory operations are performed, because the order is determined statically in the schedule. Due to this, finally the length of the schedule and size of the group are reduced and hence complexity and hardware usage reduced. The next step is, the permutation design and corresponding hardware to perform the permutation only. It means the number of locked bits impacts the 3D rotation. The placing of the bits should end up where the locked bits are located as low as possible, and preferably in the location to where we have to move the locked bits in the auxiliary permutation circuit after the memory. Otherwise, no auxiliary permutation is required for accessing the memory. Now, the counter bits are directly applied on memory according to the inverse permutation. We will need as many mappings as indicated by the *periodicity* of the required permutation. The last step is the auxiliary circuit design for permutation is based on the number of locked bits.

IV. IMPLEMENTATION

This section presents the implementation of the proposed 3D FFT, permutation circuits on FFT [4] and the transposition technique for FFT [3][10].The overall architecture of the system is shown in figure 2. This figure also shows the system modules and indexed bits and their order changing. In the any module the input and output

DOI: 10.32377/cvrjst1005

order is changed according to the order of the bits in the subs section of the module. By this, we can understand what is going on in the module. Based upon the inputs and outputs, all the modules are observed and duplicate things are ignored. The above process actually starts with the block FFT , then the process continues with the transposition and bit reversal in BRAM. This BRAM is followed by BDP circuits and it is continued with the 3D rotation in SDRAM. The auxiliary circuit for the required permutation circuit moves the locked index bits to finish the 3D rotation. The hardware usage and performance results are described below.

The 3D FFT with size of 2563 samples consists of indexed bits of 24 and 8- bits for dimension. This example is because it does not require 3d data set in cubic form and it is an easier case to understand the design. The above design is implemented on the DE3 board with the combination of a Stratix III FPGA including DDR2 of SDRAM slot. The goal of this paper is the designing of a system to give throughput of 200 M Samples per second and 1 sample per each clock cycle at the frequency of 200 Mhz.

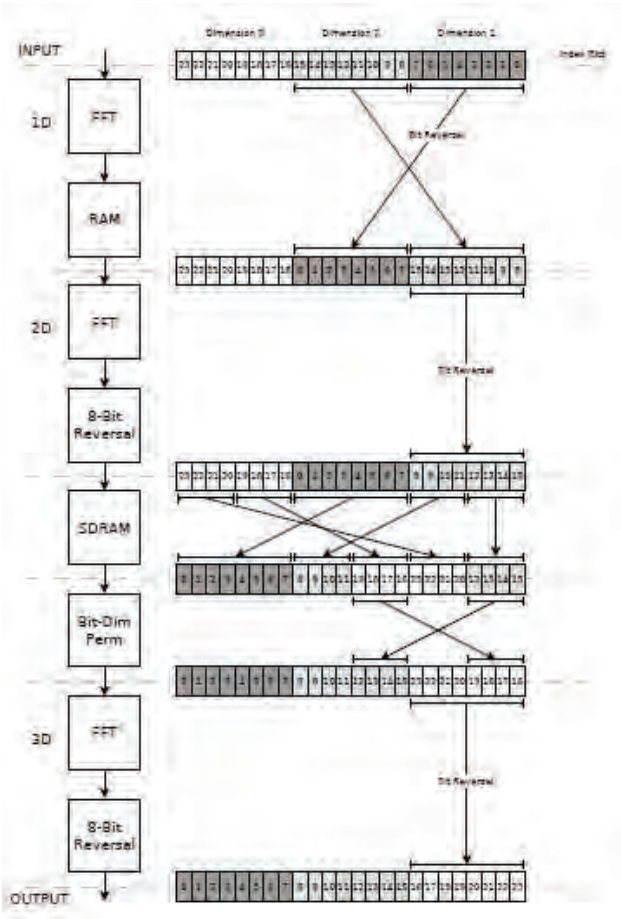

Figure 2.The system overview with bit positions of index at the input and output.

A. FFT

The placement of all identical FFT modules in the system are shown in the Figure 3.The 256-point feed-forward pipelined architecture considered. The multiplications approximation done by 14-bit precision CORDIC algorithm. From [1] the FFT module design is taken.

In the above figure, numbers specify the index bits and arrows gives the moment of the index bits done by the permutation. The shading portion in the above figure indicates the bit dimensions.

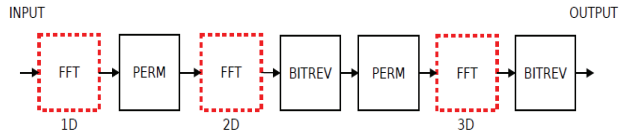

Figure 3. The FFT module highlighted system.

B. Transposition and Bit Reversal on BRAM

We need to perform two permutations between the first and second FFT Between the first and second FFT shown in figure 4. First permutation is bit reversal on the lowest eight bits for achieving natural order in the frequency domain, Second step is the transposition of matrix rows data in the second FFT. The next operation step is switching of index places for index bits on the one and two dimensions.

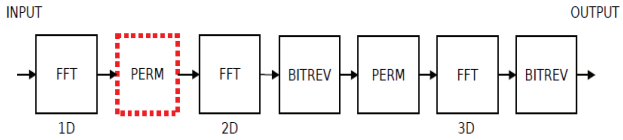

Figure 4. BRAM highlighted system overview.

For the saving of the resources, the two permutations are combined into one permutation performed on memory. This combining of permutations is easy because the BRAM did not have any access limitations and constraints. That means the read and write operations can take place any time. One and only limitation with respect to read and write operations is same location does not use double.

C. Bit Reversal

The following figure shows the bit dimensions and corresponding bit reversal circuits of the given design. This design performs the permutations using the minimum number of logic and memory resources with low complexity. This system is proposed in[4], which is desirable for efficient hardware design.

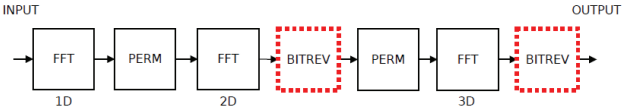

Figure 5. Architecture with bit reversal modules.

The bit reversal in FFT makes the samples in natural order with respect to frequency domain. After each FFT, the index bits reversed in the corresponding dimension is called bit reversal. In FFT architecture two bit reversal modules are included for permutations on memory. After the memory permutation for the first FFT, the transposition takes place. The auxiliary permutation circuit size is

increased in the SRRAM due to the placing of the bit reversal blocks for permutations in SDRAM. The size difference plays important role and bit reversal circuit sizes also. In the upcoming section, the sizes are compared with respect to SDRAM permutations.

D. Permutations on SDRAM

This section gives the design decisions and the corresponding implementations on the SDRAM memory. This is important criteria in the design. The following steps give the required permutations

Step 1 - Need of External Memory

Step 2 - External Memory Constraints and Parameters

Step 3 – Scheduling

Step 4 - SDRAM Permutation Design

Step 5 - Auxiliary Permutation Circuit.

Figure 6 and figure 7 shows the placing of the PERM block and the order of the input and output bits of the SDRAM respectively. Figure 8 presents the auxiliary circuit and corresponding input and output order with respect to permutations.

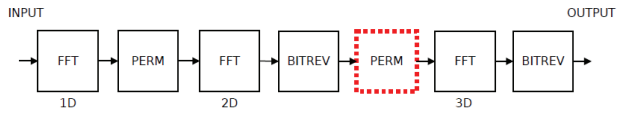

Figure 6. The SDRAM architecture and correction of the permutation blocks in the FFT system.

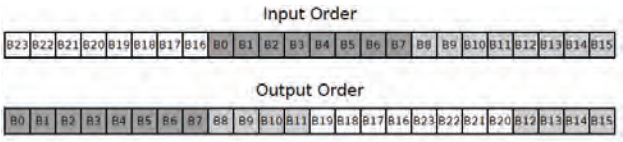

Figure 7. The input and output bit order in the SDRAM

The auxiliary permutation circuit permutes the last or lowest locked bits 19 to 16.

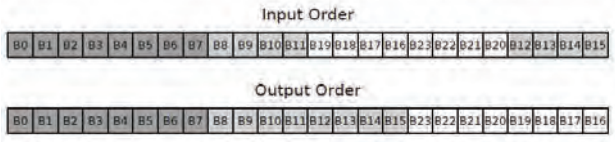

Figure 8. The input and output bit order of the auxiliary permutation circuit.

V. RESULTS

This section describes the key performance numbers in real-time systems, namely the latency and throughput. Latency, describes how long it will take for a value to propagate through the system. The throughput defines how many samples you can calculate per time unit; Based on this value you can then determine how many 3D FFTs you can calculate per time unit. The first 3D FFT calculation will be finished after the size of the data set divided by the throughput in addition to the latency. After that you will have one calculation finished after each following size of the data set divided by the throughput.

A. Throughput

In this system the system clock frequency is equal to throughput of the system. For the given system, the clock frequency is considered as 200MHz then the throughput is 200 mega samples per second. In this paper a 2563 cubic data samples are considered as example. The throughput for this example will be 11.9 frames per second, which is 84 milli seconds

B. Latency

The latency is determined with respect to number of implementation blocks in the system. The total latency is calculated using the propagation time of the samples with the aid of a counter. Final latency is determined using clock cycles. If the clock frequency is 200MHz then t_{lat} (latency) is given by

$t_{lat} = N \text{ cycles} * t_{clk} = 16848123.5 \text{ ns} = 0.084240615 \text{ s} = 84.2 \text{ msec}.$

C. Hardware Utilization

The size of the hardware depended on some factors. Mainly the design requires very high memory and logic registers of both the memories SDRAM and BRAM. The controlling of physical communication between memories, the corresponding delays and signals gives the utilization factor of SDROM. For this a Nios II soft processor is used as controller .This controller depends on utilization of on chip memory and utilization of logic. The on chip memory means the memory required for the matrix transposition. The remaining blocks are 3 FFT hardware blocks and permutation circuits. These permutation circuits consist of large number of memory elements, which occupies more logic registers since on logic blocks are used. Hence the calculation of FFTs is simple without any complex multiplications. The synthesis summary report and utilization of hardware presented in table I.

Table I. Synthesis results for hardware utilization on Altera Stratix III FPGA.

Family	Stratix III
Device	EP3SL150F1152C2
Logic utilization	28 %
Total block memory bits	2,548,445/5,630,976 (45%)
Memory ALUTs	2,595 / 56,800 (5 %)
Combinational ALUTs	19,955 / 113,600 (17 %)
Total registers	27048
Total pins	148 / 744 (20 %)
Total PLLs	1 / 8 (13 %)
DSP block 18-bit elements	0 / 384 (0 %)
Dedicated logic registers	26,870 / 113,600 (24 %)
Total DLLs	1 / 4 (25 %)

DOI: 10.32377/cvrjst1005

VI. CONCLUSIONS

This paper presented the 3D FFT algorithm design and implementation for continuously flow of samples using pipelined architecture. This paper is aimed for many real time applications. The FFT architecture is mainly based on series of three 1D FFT pipelined permutations. such a way that the data arrives in correct order to all three 1D FFTs, and that the result is in natural order in the frequency domain if this is desired.

This proposed and designed method can be used for any type memory system architectures, with or without access limitations and constraints. This implementation required two memories are on chip BRAM and external SDRAM.

This implementation is good competition for all the existing FFT designs in terms of cost , hardware, memory and performance. The further improvements in this design are including parallel dimensions in the permutations. Due to this modification, the fast memory using is possible by 1 D FFTs and then the throughput of the system is improved. It would be good to have the possibility to connect the design to a data bus and through it, a shared memory. The samples could be fetched from the data bus, and the shared memory could also be used for permutations. This would probably give a lower performance, but it would give a better opportunity to include the architecture in a complete System-on-Chip solution, and be more integrable in larger systems.

REFERENCES

[1] R. Andersson. FFT hardware architectures with reduced twiddle factor sets. Master's thesis, Department of Electrical Engineering, Linköping University,2013.

[2] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. Math. Comp., 19, 1965.

[3] M. Garrido. Efficient Hardware Architectures for the Computation of the FFT and other related Signal Processing Algorithms in Real Time. PhD thesis, Universidad Politechnical de Madrid, Spain, 2009.

[4] M. Garrido, J. Grajal, and O. Gustafsson. Optimum circuits for bit reversal. Circuits and Systems II: Express Briefs, IEEE Transactions on, 58(10):657–661, 2011.

[5] G. Halcrow and B. Mulgrew. "SAR 3D scene reconstruction using Fourier imaging techniques" in High Resolution Imaging and Target Classification, 2006. The Institution of Engineering and Technology Seminar on, pages 53– 60, 2006.

[6] A. Kojima, N. Sakurai, and J. I. Kishigami. Motion detection using 3DFFT spectrum. In Proc. IEEE Int Acoustics, Speech, and Signal Processing ICASSP-93. Conf, volume 5, pages 213–216, 1993.

[7] U. Nidhi, P. Kolin, H. Ahmed, and K. Anshul. High performance 3D-FFT implementation. In Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, pages 2227–2230, 2013.

[8] C.-L. Yu and C. Chakrabarti. Transpose-free SAR imaging on FPGA platform. In Proc. IEEE Int Circuits and Systems (ISCAS) Symp, pages 762–765, 2012.

[9] C.-L. Yu, K. Irick, C. Chakrabarti, and V. Narayanan. "Multidimensional DFT IP generator for FPGA platforms" Circuits and Systems I: Regular Papers, IEEE Transactions on, 58(4):755–764, 2011.

[10] B. Akin, P. A. Milder, F. Franchetti, and J. C. Hoe. "Memory bandwidth efficient two-dimensional fast Fourier transform algorithm and implementation for large problem sizes" In Proc. IEEE 20th Annual Int. Field-Programmable Custom Computing Machines (FCCM) Symp, pages 188–191, 2012.