# Fast Convolution with a PN Code Based on Field Programmable Gate Array

[1]Prof. J.Mehena [2] Lopamudra Swain, [3] B.P. Mishra
Department of Electronics & Telecommunication Engg.
DRIEMS,Tangi,Cutttack

**Abstract**

**In code division multiple access (CDMA) system, receivers spend long time to acquire the signals. This is mostly due to the use of expensive FFT-based convolvers in the acquisition process. This paper shows a substitute algorithm for calculating the convolution that requires less computation time. The algorithm uses Walsh transform instead of FFTs. FFT based algorithm requires 2 FFTs and one IFFT in addition to complex multiplications and additions. On the other hand, in the Walsh-based method the Walsh transform is required once and there is no multiplications. Therefore, using the Walsh-based algorithm can cut the processing time to about 5 percent of the required time. The additional steps in this algorithm are the permutation of the input samples and the output results. The design uses a field programmable gate array (FPGA) to apply parallel processing concept. This paper discusses the algorithm and the implementation issues. A case study of a large code was applied. The whole system is implemented and showed high performance that speedup the process 2500 times the speed of a microprocessor based design.**

**KEYWORDS:** CDMA, FFT, IFFT, FPGA, Walsh Transform, Convolution.

## I. INTRODUCTION

In a code division multiple access (CDMA) system, information is modulated with a pseudo random noise (PN) code. The used PN codes have autocorrelation property only with a synchronized copy of itself [1-2]. A receiver should generate a synchronized copy of the code and multiply it by the received signal. If the local code is perfectly synchronized, then its correlation with the original signal is maximum; otherwise the correlation is very low. Synchronizing a CDMA signal with local code is called acquisition. Acquisition, in GPS system for example, is used to calculate the code phase (or shift) and find the pseudo range, which is used to calculate the position. Acquisition of the signal in GPS receivers is the most critical part due to the required number of processing steps and long processing time. It is important to perform the acquisition as fast as possible in order for auto navigation system to be practical [2-3].

Recent research has implemented the acquisition process using the circular convolution that can check all the phase shifts in one step. This is usually done by using the known FFT-based convolution (Fig.1). Each FFT (or IFFT) requires NlogN complex multiplications and NlogN complex additions. Therefore, this algorithm requires approximately 3N(logN)+N complex multiplications and 3N(logN)+N additions [4]. Implementing the algorithm in a parallel hardware will speed up the process but the implementation itself is very complex and requires a huge silicon area. To

alleviate these difficulties, a new algorithm for fast convolution is proposed in this paper. A binary transformation (such as Walsh transform) is suggested since it requires only NlogN additions and subtractions. Finding a method to perform the convolution using only Walsh transforms and then implementing it using parallel processing units such as those in the FPGAs will definitely speed up the acquisition process.

In the next sections, a Walsh-based convolution algorithm is presented. First, the similarity between the Walsh transform and the PN sequences is described. Then, the Walsh-based convolution algorithm is provided. This algorithm requires functions that need to be implemented efficiently in hardware to build a very fast convolver. The

hardware implementations of the main functions are presented next. Then a discussion of the design and its results is provided. Evaluations of the design performance, based on clock frequency and system latency, are presented.

 

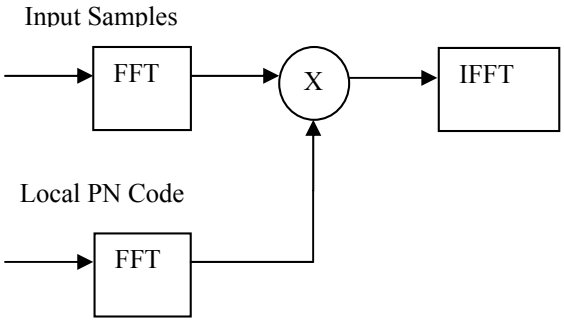


Fig.1 FFT based Convolver

## II. PN SEQUENCES AND WALSH TRANSFORM

The Walsh function can be transformed into a set of different phase shifts of a single PN sequence using suitable permutations [5-6]. This can be noticed if the

first row and the first column of a Walsh matrix are omitted. Permuting the remaining rows and columns using specific permutation method will produce a matrix with a PN code in the first row and all the shifted copies of this code are surprisingly the remaining rows [ 6-7]. This is a useful property since all the PN code shifts are available in one matrix. If a received signal contains this PN code, then it is clear that multiplying the signal by the obtained matrix will generate a vector that

has a large multiplication value only with one of the PN shift copies (or Walsh rows). This means that the Walsh transform can be used to implement convolution with a PN sequence [7].

## III. CONVOLUTION ALGORITHM

The proposed convolution algorithm requires two additional steps in order to use Walsh transform to implement convolution. The complete procedure, which includes these steps, is as follows:

1- The input sequence must be permuted to simulate reordering of rows.

2- The Walsh function is applied.

3- After that, the results will need to be permuted back to correctly reorder the convolution values to return back to their original locations.

These two steps of permutation processes require two different permutation vectors, S and C [7]. As an example, a PN code with a period of 7 is used to simplify the description of the algorithm. This example is presented in details in [7]. An 8-point Walsh transform is used for the 7 bit-length PN code. After omitting the first row and the first column, the Walsh transform becomes a 7-point transform. For the PN code such as (1001110), the input permutation S is (1,4,6,7,3,5,2) and the inverse permutation of S is (1,7,5,2,6,3,4). The output permutation Q (or C as in [7]) is (1,2,4,5,7,3,6). To perform correlation using this method, first the input sequence is permuted by permutation of inverse S. Since the Walsh is larger than the sequence by one, appending (0) to the beginning of the permuted sequence must be made. After that, Walsh transform is applied. The results are not in the order they should be in Qs compared to the convolution results. Therefore, the output should be reordered based on permutations Q [7]. This method requires only NlogN real additions (and subtractions). However, the FFT-based method requires 3 FFTs and N complex multiplications and N complex additions. If the PN code FFT was stored in the computer memory, then only 2 FFTs plus N complex multiplications and additions are required with the incoming signal. This is approximately 2NlogN+N complex multiplications and 2NlogN+N complex additions, which is roughly about 6NlogN real multiplications and 6NlogN real additions [4,8]. Therefore, the Walsh-based correlation method is preferable especially in real-time applications, where the acquisition process needs to be very fast. More details about the Walsh-based convolution algorithm and its in-depth theory can be found in [7,9].

## IV. IMPLEMENTATION

The main steps that need to be carefully implemented are the permutation generators and the Walsh transform. The permutations usually can be stored in lookup tables (LUTs). This type of implementation is not efficient since it will cost additional hardware to store and time to retrieve. Therefore, the permutations need to be generated on fly when possible to minimize the required silicon area and to speedup permutation process. However, the implementation of Walsh transform must consider two requirements. A Walsh transform should use the parallel processing method as much as possible and chose optimum smaller transform block sizes for building very large transforms.

### *A. Permutation Generators*

PN sequences are generated using linear feedback shift registers (LFSRs) [2,5-7,10]. If the permutations can be related to the state of the LFSR, then the permutations can be

generated from the same LFSR. This is true for permutations S, which are the decimal values of the binary bits in the register. When initializing the LFSR with (001) for the previous example, the current state and the next six states (or values) of the register will be (001, 100, 110, 111, 011, 101, 010), or (1, 4, 6, 7, 3, 5, 2) in decimal which is the required permutation sequence S. Therefore, the permutations S can be generated easily. A hardware implementation of generator of

permutations S is shown in Fig. 2. The RAM and the counter are needed only when storing permutations is desired.

However, in the correlation algorithm, the only inverse permutation of S is used, therefore no hardware is used to store S. The hardware implementation of S inverse permutations generator is shown in Fig.3. The only change made to Fig.2 is that the LFSR is used to deliver RAM

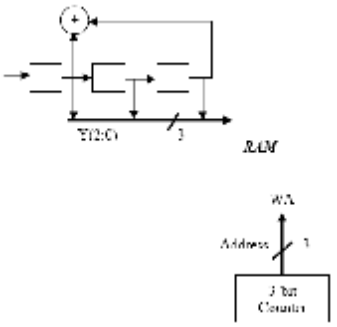


Fig. 2. Generator of permutations S

address while counter output is fed to RAM as data in. As mentioned before, storing permutations is not efficient, but since generating inverse permutations of S is required only once in the beginning of the design, this

implementation is accepted. Permutations Q fortunately, as well, are related to the content of the LFSR.

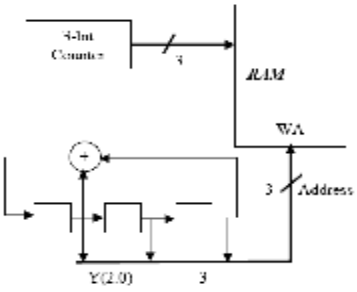


Fig.3.Generator of inverse permutations of S

Each PN code has its unique permutation of LFSR bits to produce Q. Reordering the content of the LFSR (i.e. b2b1b0 becomes b2b0b1) will generate the necessary Q sequence. On other words, reordering the bits of the

content of the LFSR can be implemented using one-to-many type of LFSR as shown in Fig.4 [5].

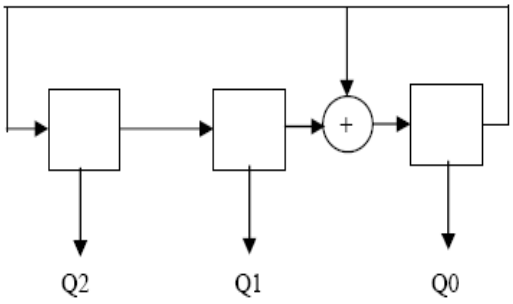


Fig. 4. Generator of permutations Q

*B. Walsh Transform*

Designing transforms in parallel processor platform such as field programmable gate arrays (FPGAs) is recommended since transforms need large number of operations that can be easily mapped into FPGAs. Hardware design of Walsh transform using butterfly structure is very efficient because of many reasons. The most important reason is that Walsh butterfly has only one type of operation which is the add operation (subtractions are similar to additions) as shown in Fig.5. Therefore, when a designer wants to partition the Walsh butterfly, all what he cares about is the locations of the inputs, intermediate values, and outputs. The fastest Walsh butterfly should be implemented in parallel fashion to reach the maximum speed. Unfortunately, for DSP applications, Walsh size can be huge. Since the number of processors required to implement a completely parallel transform is very large and requires large number of input and output pads. Therefore, a completely parallel design is impossible since there is no chip that can support these requirements. A solution to this problem would be partitioning the butterfly into smaller butterflies. The size and the necessary number of smaller butterflies will play a role in the design organization and performance of the Walsh transform chip. For a 1024- point Walsh butterfly, a designer can use partition of 128, 64, 32 or 16 –point butterflies for example. 128-point butterfly may not be a good choice because 8-point butterflies will be also required to build the 1024-point Walsh transform. Therefore, choosing a different size of butterfly could help in reducing the number of blocks and will help in designing a well organized structure. Another factor that controls the design is, the available resources in board or chip. This means that even if the designer found an optimum size of smaller butterflies, he may need to split it too in smaller butterflies

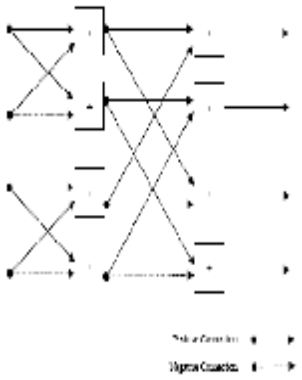

Fig. 5. Walsh butterfly

because of the lack of resources. Using large FPGAs such as the Virtex FPGA series will provide more resources for designs as large as 1024-point transforms. For 1024-point Walsh butterfly, 64 blocks of 32-point Walsh butterflies is the optimum solution if the 32-point Walsh block is designed completely in parallel method. This is explained in the next section.

## V. CASE STUDY

To implement a Walsh-based convolver, a PN code of period 1023 is used. A 1024 point Walsh transform is necessary to implement the convolution. The optimum size of smaller blocks of butterflies is chosen to be 32-point. This is because 32 point Walsh butterfly needs to be processed 64 times and there will be no other smaller

size of butterflies required. This is not the case if different size was chosen for the implementation. Since the platform is the 0.8 million gate FPGA (Virtex), the 32-point butterfly is implemented in parallel and used to calculate the first 5 levels of the 1024 butterfly. A similar block is also used to calculate the second 5 levels of the 1024 butterfly (Fig. 6). The second block was used to provide a pipelined type of system that reduces the latency of the system compared to a system with only one 32- point Walsh butterfly and one RAM. The completed design used more components to be able to provide updated evaluation of phase shifts every 3 code lengths (3069 clock cycles). These components are required to perform Walsh transform computations, permutations generation, correlation peak search, RAMs to store intermediate butterfly results, counters for RAM addresses, and state machines to control the whole system to continuously provide phase information in constant time space. The whole design used about 60% of the Virtex chip (approximately 264K gates). The maximum frequency for this design is 96 MHz. The results showed that this design can be used for real time processing when skipping 2 code periods and loading the next period-length information is acceptable.

To evaluate the performance of this design compared to FFT-based design, an assumption needs to be made. If we assume that a multiplication takes as much as two additions, then the total number of operations in FFT-based method is about 18 times larger than the Walsh-based algorithm. This means that this algorithm reduced the processing time to 1/18 (about 5%). In addition an N-bit multiplier takes N times larger area than N-bit adder, so for instance if input signals are 8-bit long, then the time area efficiency of this algorithm is about 144 times better than FFT-based method. This is in case the whole transforms are computed completely in parallel. However, this is not the case. Therefore, design analysis within limited area is more appropriate. Let assume

that the design is limited to area necessary for 32-point Walsh butterfly. Then, using Walsh-based method, the convolver of 1023-length PN code requires 64 blocks of 32-point Walsh transforms. Therefore, it will take 64*32 (=2048) clock cycles. While on the other hand, the maximum size of FFT

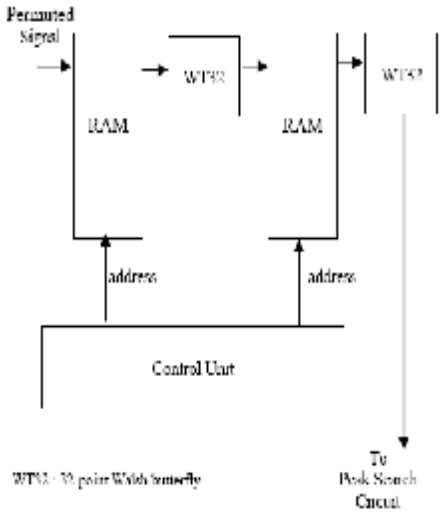


Fig. 6. Simplified block diagram of the 1023bit PN sequence convolver

butterfly that can be mapped to the limited area is only 4- point FFT. This means that 4*256*5 (=5120) clock cycles are necessary to compute a 1024-point FFT. In addition, in each clock cycle required in 32-point Walsh butterfly we need to perform 5 consecutive additions, while in 4-point FFT we need to do 6 multiplications and 10 additions. Relatively, 1024-point FFT requires about 10 times longer than a 1024-point Walsh transform needs. Since FFT-based convolver needs two FFTs, the minimum time required for FFT-based convolver is about 20 times larger than what is required for Walsh–based method. In other words, Walsh-based method is at least 20 times faster than FFT based method when the design area is limited (for instance in Virtex FPGA implementation). When comparing this implementation with a software-based implementation of the Walsh transform on

Matlab on PC, the FPGA-based implementation was about 2500 faster than a 233MHz processor. This is only using a 1MHz clock in the FPGA design. To speed up the processing time more, another fast clock can be used with the internal processing parts and using multiplexers to switch between the two clocks. For example, if the loading of incoming samples uses 1MHz of clock frequency, then we can speedup the convolution process by allowing the internal parts of the design (such as reading from memories, 32-point Walsh transform, and finite state machines) to use different clock frequency such as 16MHz. This insures that the system can be used efficiently and the design can provide new phase shift every 2 code periods instead of three code periods. This may cause synchronization problems and therefore it will need a careful study.

## VI. CONCLUSION

This paper presented a design of fast convolver for CDMA signals. This is based on avoiding complex operations such as the ones in FFT-based convolvers. The substitute of the FFT is a binary transform, such as Walsh, that should reduce the operations 3 times because it uses only real

additions. Surprisingly, the used algorithm does not require using the transform more than once. This made the method more efficient. The Walsh function can be transformed into a set of different phase shifts of a single PN sequence using suitable permutations. The convolution can be then performed by reordering the input sequence, performing the Walsh transform, and then permuting the output samples. This method beats FFT-based convolution since it requires NlogN additions while FFT method requires about 6NlogN multiplications and 6NlogN additions. For a PN code of a period length equal to 1023, a 10-bit LFSR is used to generate the permutations in real time. This is more efficient than storing them in Lookup Tables. A Walsh transform can be implemented in an FPGA using butterfly structure. However, since it is impossible to perform the Walsh transform of all the input samples completely in parallel, partitioning the method into smaller size blocks was used. These smaller blocks use parallel architecture to speed up the operations. On the other hand, each time the Walsh Transform is calculated for a single block, the result values need to be stored sequentially into local memories. The size

of the used FPGA and the size of the blocks played the main role in the overall performance of the algorithm. The implemented design has speeded up acquisition of CDMA signals many times, and opened new research topics for the applications where real-time acquisition is needed.

## VII. REFERENCES

[1] F. Mazda, Telecommunications Engineer's Reference Book ,2nd edition, Focal Presss,1998, p. 27/13.
[2] E. D. Kaplan, Understanding GPS Principles And Applications, Artech House Inc., MA: 1996, p. 100-104.
[3] M. Kayton, W. Fried, Avionics Navigation Systems ,2nd edition, J. Wiley & Sons Inc,1997.
[4] W. W. Smith, J. M. Smith, Handbook Of Real-Time Fast Fourier Transforms, IEEE Press, 1995, p. 28.
[5] M. Cohn, A. Lempel, "On fast M Sequence transforms", IEEE Trans. On IT, Jan. 77, p. 135-137.
[6] A. Lempel, "Hadamard and M-Sequence transforms are permutationally similar", Applied Optics, Vol. 18, No. 24, 15 Dec 1979, p.4064-4065.
[7] Srdjan Z. Budisin, "Fast PN Sequence Correlation By Using FWT," Mediterranean Electrotechnical Conference Proc., 1989, p. 513-515.
[8] M. Uijt de Haag, "An Investigation Into The Application of Block Processing Techniques For The Global Positioning System", Ph.D. Dissertations, Ohio University, 1999.
[9] K. G. Beauchamp, Applications Of Walsh And Related Functions, Academic Press Inc., 1984.
[10] S. W. Golomb, Shift Register Sequences, Holden-Day Inc., 1967..

 