International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-6, Issue-8)

Research Article | **August 2017**

Design and Implementation of Fast Fourier Transform (FFT) using VHDL Code

Akarshika Singhal, Anjana Goen, Tanu Trushna Mohapatrara
Department of Electronics and Communication, Rustamji Institute of Technology, B.S.F Academy Tekanpur, Gwalior, Madhya Pradesh, India

Abstract:

The Discrete Fourier Transform (DFT) can be implemented very fast using Fast Fourier Transform (FFT). It is one of the finest operation in the area of digital signal and image processing. FFT is a luxurious operation in terms of MAC. To achieve FFT calculation with a many points and with maximum number of samples the MACs requirement could not be matched by efficient hardware's like DSP. A parallel and pipelined Fast Fourier Transform (FFT) processor for use in the Orthogonal Frequency division Multiplexer (OFDM) and WLAN, unlike being stored in the traditional ROM. The twiddle factors in our pipelined FFT processor can be accessed directly. In this paper, we present the implementation of fast algorithms for the DFT for evaluating their performance. The performance of this algorithm by implementing them on the Xillinx 9.2i Spartan 3E FPGAs by developing our own FFT processor architecture.

Keywords: Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Field Programmable Gate Array (FPGA), Multiplier - Accumulator (MAC), Orthogonal Frequency Division Multiplexing (OFDM), Wireless Local Area Network (WLAN)

I. INTRODUCTION

In the recent decades DFT has been playing several important roles in advanced applications such as image compression and reconstruction in biomedical images, audiology research for analyzing biomedical brain-stem speech signals, sound filtering, data compression, partial differential equations, and multiplication of large integers.

The fast algorithms for DFT always look for DFT process to be fast, accurate and simple. Fast is the most important [5]. Since the introduction of the Fast Fourier Transform (FFT), Fourier analysis has become one of the most frequently used tool in signal/image processing and communication systems; The main problem when calculating the transform relates to construction of the decomposition, namely, the transition to the short DFT's with minimal computational complexity. The computation of unitary transforms is complicated and time consuming process. Since the decomposition of the DFT is not unique, it is natural to ask how to manage splitting and how to obtain the fastest algorithm of the DFT. The difference between the lower bound of arithmetical operations and the complexity of fast transform algorithms shows that it is possible to obtain FFT algorithms of various speed [2]. One approach is to design efficient manageable split algorithms. Indeed, many algorithms make different assumptions about the transform length [5]. The signal/image processing related to engineering research becomes increasingly dependent on the development and implementation of the algorithms of orthogonal or non-orthogonal transforms and convolution operations in modern computer systems. The increasing importance of processing large vectors and parallel computing in many scientific and engineering applications require new ideas for designing super-efficient algorithms of the transforms and their implementations [2].

The hardware implementation of FFT approaches is a challenging issue where the digital signal processors (DSPs) and the field programmable gate array (FPGA) chips are two considering designing environments for implementing different schemes of FFT approaches.

Recently, the FPGA technology [3] is quit mature for digital signal processing applications [4] due to fast progress in very large scale integration (VLSI) technology. The FPGA devices provide fully programmable system-on-chip environments by incorporating the programmability of programmable logic devices and the architecture of gate arrays. They consist of thousands of logic gates and some configurable logic blocks which make them an appropriate solution for prototyping the application specific integrated circuits (ASIC) with dedicated architectures for specified digital signal processing applications. The introduction of Verilog Hardware Description Language (HDL) [5] provided a modeling and simulation environment for fast prototyping digital circuits and systems on FPGA.

Implementing of different schemes of FFT algorithms and applications received much attention in literature [6-10]. The aim of this paper is to model and hardware description of different schemes of FFT approaches including Cooley-Tukey, Good-Thomas, Radix-2 and Rader methods by Verilog HDL and realization of them on Xilinx FPGA chip. Then the performance of different algorithms is compared for chip area utilization and critical path time.

The rest of the paper is as follows; Section 2 describes four well-known FFT approaches using mathematical models and block diagrams. FPGA implementation of FFT approaches and comparing their performance are presented in section 3 and finally the paper is concluded in section 4.

II. FFT ALGORITHMS

In this section four common FFT methods including Cooley-Tukey, Good-Thomas, Radix-2 and Rader are described in details and the mathematical models of them are reviewed. For this, the mathematical background of each method is presented and the block diagram of each approach for N-point FFT operation is provided.

A. The Cooley-Tukey FFT Algorithm

This method was proposed by Cooley and Tukey [11] and generally is used for computing FFT. In this approach the number of FFT points can be divided into two factors [12], N_1 and N_2 as follows;

$$N = N_1 \times N_2 \quad (1)$$

Input indexes, n, are obtained from following expression;

$$n = N_2 \times n_1 + n_2 \; where \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases} \quad (2)$$

Furthermore, the output indexes, k, are obtained from following expression;

$$k = k_1 + N_1 \times k_2 \; where \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases} \quad (3)$$

For example, when N is 15 then N_1 and N_2 are chosen to be 3 and 5 respectively. According to (2) and (3) input and output indexes are described by following expressions;

$$k = k_1 + 3k_2 \; where \begin{cases} 0 \le k_1 \le 2 \\ 0 \le k_2 \le 4 \end{cases} \quad (4)$$

$$n = 5n_1 + n_2 \; where \begin{cases} 0 \le n_1 \le 2 \\ 0 \le n_2 \le 4 \end{cases} \quad (5)$$

The block diagram of this method for N = 15 is shown in fig 2.

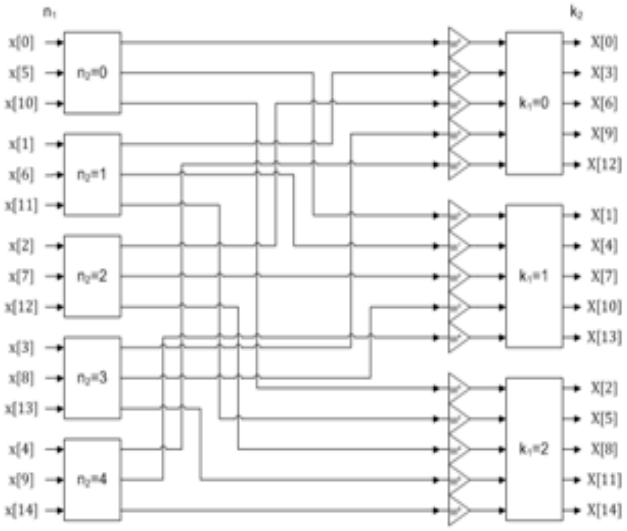

Fig.1. The block diagram of 15 point Cooley-Tukey Algorithm

B. The Good-Thomas FFT Algorithm

This method was suggested by good [13] and Thomas [14]. Considering Cooley-Tukey method shown in Figure 2, between two blocks at the front end and back end, some coefficients are placed. These coefficients can be eliminated by some assumptions and consequently for the same number of FFT points this method has less chip area occupation. Main supporting idea of this method is that N_1 and N_2, those are two-factor of N, are prime to each other. Input and output index mapping is done according to equation (6) and (7), respectively.

$$n = N_2 n_1 + N_1 n_2 mod\ N \; where \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases} \quad (6)$$

$$k = N_2 . Ak_1 + N_1 . Bk_2 mod N \; where \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases} \quad (7)$$

AN_2 and BN_1 satisfy following equations:

$$AN_2 \times N_1 mod\ N = 0 \quad (8)$$

$$BN_1 \times N_2 mod\ N = 0 \quad (9)$$

Furthermore, in Good-Thomas method, in addition to equations (8) and (9), below assumptions must be established at the same time:

$$BN_1^2 mod\ N = N_1 \quad (10)$$

$$AN_2^2 mod\ N = N_2 \quad (11)$$

For N=15, $N_1$and N_2 are 3 and 5 respectively and input and output indexes are:

$$n = 5n_1 + 3n_2 mod\ 15 \; where \begin{cases} 0 \le n_1 \le 2 \\ 0 \le n_2 \le 4 \end{cases} \quad (12)$$

$$k = 10k_1 + 6k_2 mod 15 \; where \begin{cases} 0 \le k_1 \le 2 \\ 0 \le k_2 \le 4 \end{cases} \quad (13)$$

The block diagram of Good-Thomas method for N= 15 is presented in Figure 2.

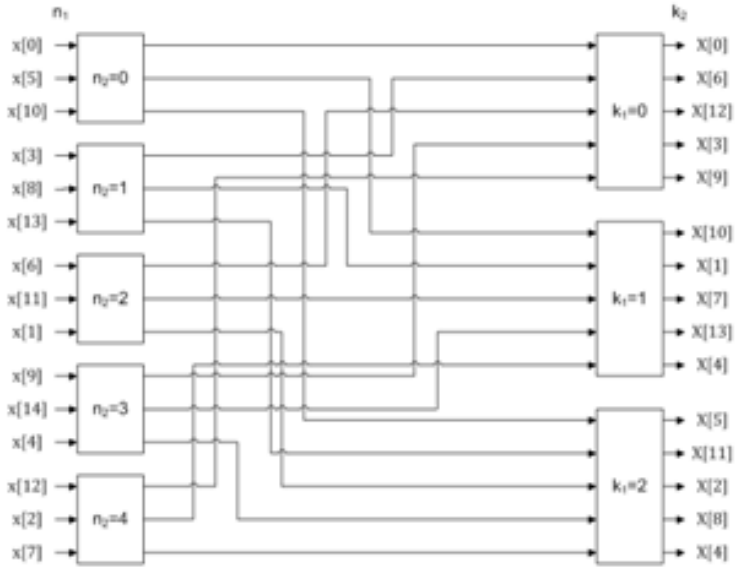

Fig.2. The block diagram of 15-point Good-Thomas FFT Algorithm

C. The Radix -2 FFT Algorithm

This method is the subset of the Cooley-Tukey method. In this method, N_1 or N_2 is chosen to be 2 and the other one is N/2. It is assumed that N is a power of 2 [15-17]. As an example, for N=16, N_1=2 and N_2 is 8 and the following equation describe the implementing approach of this method.

The main expression of the Fourier transform is (14) and N=16 then k = 0,1,2,…….,15

$$Y(k) = \sum_0^{15} y(n) W_{16}^{nk} \quad (14)$$

If odd and even parts of (14) would be separated, then (15) is:

$$Y(k) = \sum_0^7 y(2n) W_{16}^{2nk} + \sum_0^7 y(2n+1) W_{16}^{(2n+1)k} \quad (15)$$

Considering $W_{16}^{2nk} = W_8^{nk}$ then:

$$Y(k) = \sum_0^7 y(2n) W_8^{nk} + W_{16}^k \times \sum_0^7 y(2n+1) W_8^{nk} \quad (16)$$

Furthermore when $W_{16}^{k+8} = -W_{16}^k$ then:

$$Y(k) = \sum_0^3 y(4n) W_8^{2nk} + W_8^k \times \sum_0^3 y(4n+2) W_8^{(2n+1)k} + \quad (17)$$

$$W_{16}^k \times (\sum_0^3 y(4n+1) W_8^{2nk} + W_8^k \times \sum_0^3 y(4n+3) W_8^{(2n+1)k})$$

And eventually;

$$Y(k) = \sum_0^1 y(8n) W_2^{nk} + W_{16}^{4k} \times \sum_0^1 y(8n+4) W_2^{nk} +$$

$$W_{16}^{2k} \left(\sum_0^1 y(8n+2) W_2^{nk} + W_{16}^{4k} \times \sum_0^1 y(8n+6) W_2^{nk} \right) +$$

$$W_{16}^k (\sum_0^1 y(8n+1) W_2^{nk} + W_{16}^{4k} \times \sum_0^1 y(8n+5) W_2^{nk} +$$

$$W_{16}^{2k} (\sum_0^1 y(8n+3) W_2^{nk} + W_{16}^{4k} \times \sum_0^1 y(8n+7) W_2^{nk})) \quad (18)$$

The butterfly diagram of Radix-2 FFT method for N=16 is presented in Figure 3.

D. The Rader's FFT Algorithm

This method was introduced by Rader [18, 19] where it is assumed that N is prime. According to [20, 21], each prime number has one or more primitive root, here called p. Selecting one of the primitive root using (19) , the input index sequence can be obtained so that the FFT operation is calculated with the cyclic convolution . The output indexes of the convolution are 1 to N-1 and the 0 index must be calculated separately.

$$p^n mod N \quad 0 \le n \le N-2 \quad (19)$$

For example when N=17, the primitive roots are 3, 5, 6, 7, 10, 11, 12, and 14. Choosing number 3 as primitive root the order of inputs is obtained from Table1.

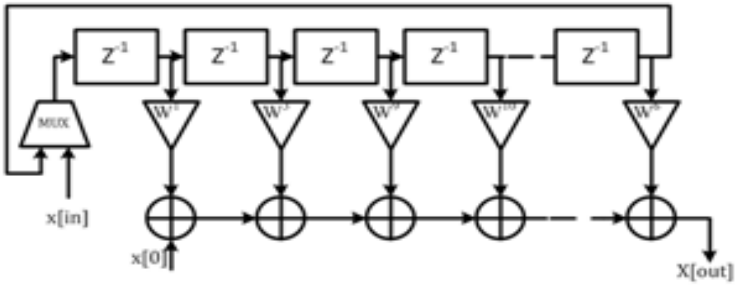

Fig. 4. Cyclic Convolution of Rader FFT Algorithm

In addition to input indexes, the coefficients must arranged like inputs indexes.

Figure 4 shows the cyclic convolution part of Rader FFT method. The entries come into the rotational part and go forward with each clock pulse.

On the other hand, these entries are multiplied by the coefficient weights and added with x[0] finally to make the output.

Table I The Index Number of Rader FFT Algorithm

n	Index number	n	Index number
0	1	8	16
1	3	9	14
2	9	10	8
3	10	11	7
4	13	12	4
5	5	13	12
6	15	14	2

III. FPGA IMPLEMENTING AND COMPARISON STUDY

We have implemented the architectures for radix-2 on Spartan 3E FPGAs. As there are embedded dedicated multipliers and embedded block RAMs available, we can use them without using distributed logic, which economize some of the CLBs. As we are having Extreme DSP slices on Spartan 3E FPGAs**,** to utilize them and improve speed performance of FFT.

In the Fast Fourier Transform process the butterfly operation is the main unit on which the speed of the whole process. The adders and subtractors are implemented using the LUTs (distributed arithmetic). The inputs and outputs of all the arithmetic units can be registered or non-registered.

We have considered the implementation of both embedded and distributed multipliers; the latter are implemented using the LUTs in the CLBs. The three considerations for inputs/outputs are with non-registered inputs and outputs, and with registered inputs and outputs. To implement butterfly operation for its speed improvement and resource requirement, we have implemented both multiplication procedure based on the availability of number of embedded multipliers and design feasibility.

The performances of FFT algorithm in terms of chip area utilization and maximum operating frequency on target chip. The FFT algorithm are modeled by VHDL and implemented on XC3S100E chip from Xillinx Inc.[14]. The specification of the test bench chip is listed in Table 2. The ISE software is used for synthesize and simulation of VHDL codes. The result of FPGA chip are occupation including number of utilized slices and Flip Flops by FFT algorithm.

Table II The Specification Of Test Bench Chip

Partition Summary	Utilizations
Slices	33260
Slice Flip Flop	66450
LUTs	66450

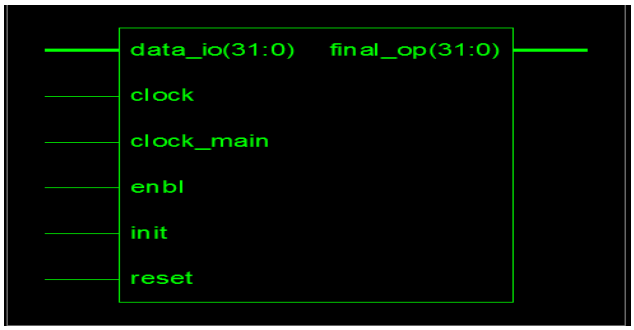

Fig.5. RTL structure of FFT

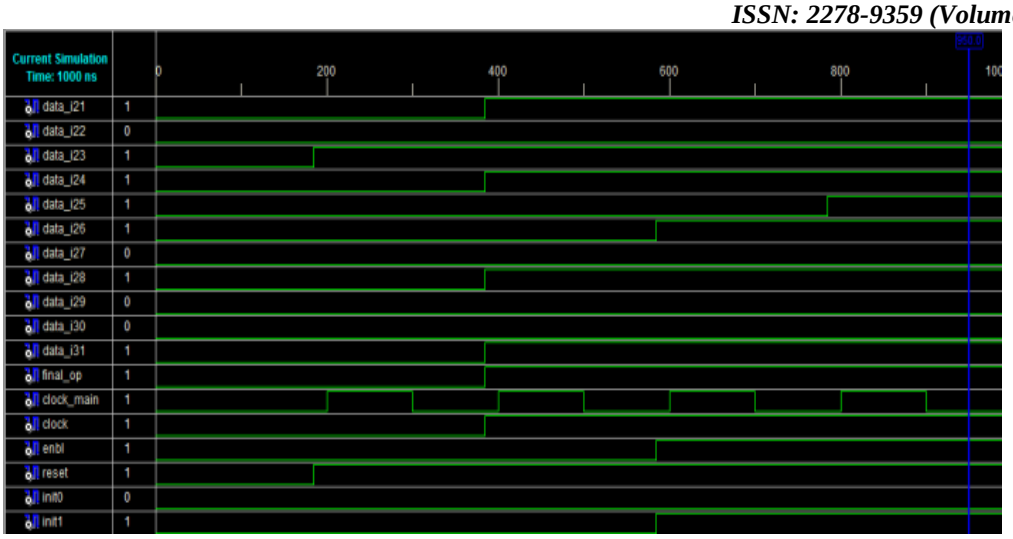

Fig.6. Test bench Waveform of FFT

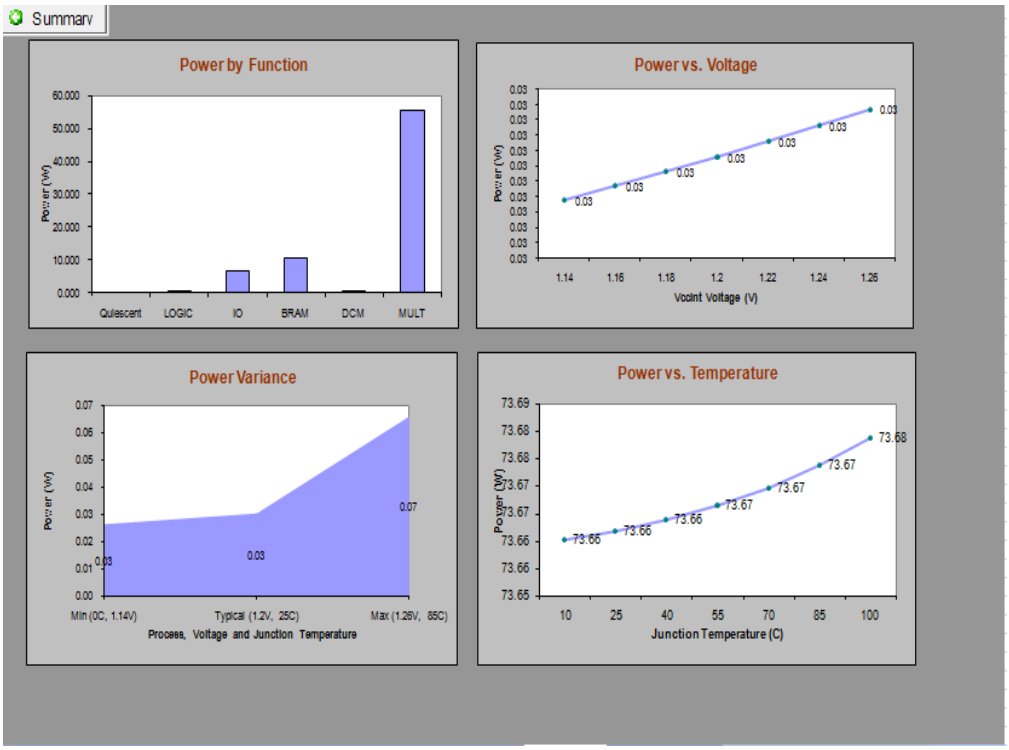

Fig.7. Power versus Function, voltage, variance and Temperature

IV. CONCLUSION

This project describes the efficient use of VHDL code for the implementation of radix 2 based FFT architecture and the wave form result of the various stages has been obtained successfully. The accuracy in obtained results has been increased with the help of efficient coding in VHDL. The accuracy in results depends upon the equations obtained from the butterfly diagram and then on the correct drawing of scheduling diagrams based on these equations.

REFERENCES

[1] S. J. Vaughan-Nichols: "OFDM: Back to the Wireless Future" *Computer*, 35(12), 19–21, (2002).
[2] K. Sobaihi, A. Hammoudeh, D. Scammell: "FPGA Implementation of OFDM Transceiver for a 60GHzWireless Mobile Radio System" International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp.185-189, 2010.
[3] W. Wolf, *FPGA-Based System Design. Englewood Cliffs*, Prentice- Hall, 2004.
[4] U. M.-Baese, *Digital Signal Processing With Field Programmable Gate Arrays*, Springer, 2007.
[5] S. Palnitkar: *Verilog HDL, A Guide to Digital Design and Synthesis*. Englewood Cliffs, NJ: Prentice-Hall, (1996).
[6] Dong-sun Kim, Seung-yerl Lee, " Dual input radix 23 SDFIFFT/FFT processor for wireless multi-channel real sound speakers using time division duplex scheme", *IEEE Transactions on Consumer Electronics,* 55(4),.2323-2328, (2009).

[7] M. A. Sanchez, M. Garrido, M. Lopez-Vallejo, J. Grajal, "Implementing FFT-based digital channelized receivers on FPGA platforms", *IEEE Transactions on Aerospace and Electronic Systems*, (44) 4, 1567 – 1585, (2008).
[8] V. Gautam, K.C. Ray, P. Haddow, "Hardware efficient design of Variable Length FFT Processor", 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 309 –312, 2011.
[9] Long Pang, Bocheng Zhu, He Chen, " Design and realization of small point FFT processor based on twiddle factor classification", International Conference on Electronics, Communications and Control (ICECC), pp.1396 – 1399, 2011.
[10] A. Ghouwayel, Y. Louet, "FPGA implementation of a reconfigurable FFT for multi-standard systems in software radio context", *IEEE Transactions on Consumer Electronics*, 55(2), 950-958, (2009).
[11] Cooley, James W., and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, 19, 297–301, (1965).
[12] Duhamel, P., and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art", *Signal Processing*, 19, 259–299, (1990).
[13] I. Good, "The Relationship between Two Fast Fourier Transforms," *IEEE Transactions on Computers*, 20, 310–317, (1971).
[14] L. Thomas, "Using a Computer to Solve Problems in Physics," *Applications of Digital Computers*, 1963
[15] P. Duhamel and H. Hollmann, "Split-radix FFT Algorithm", *Electron. Lett.*, 20(1), 14–16, (1984).
[16] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, 6(4),. 267–278, (1984).
[17] J. B. Martens, "Recursive cyclotomic factorization—A new algorithm for calculating the discrete Fourier transform, "*IEEE Trans. Acoust., Speech, Signal Processing*, 32(4), 750–761, (1984).
[18] C. Rader, "Discrete Fourier Transform when the Number of Data Samples is Prime", *Proceedings of the IEEE* 56, 1107–8 (1968).
[19] J. McClellan, C. Rader, "Number Theory in Digital Signal Processing", *Prentice-Hall Signal processing series*, 1979
[20] Bach, Eric; Shallit, Jeffrey, "*Algorithmic Number Theory* (Vol I: Efficient Algorithms)*", Cambridge:* The MIT Press, 1996
[21] Cohen, Henri , "*A Course in Computational Algebraic Number Theory*", *Springer*, 1993.

