DESIGN AND IMPLEMENTATION OF A FFT PRUNING ENGINE FOR DSA-ENABLED COGNITIVE RADIOS

Manuele Cucchi , Deepak Revanna , Roberto Airoldi and Jari Nurmi
Tampere University of Technology
Department of Electronics and Communications Engineering
P.O. Box 553, FIN-33101, Tampere Finland
name.surname(at)tut.fi

ABSTRACT

This research work presents the design and implementation of an FFT pruning block, as an extension of an FFT core for OFDM demodulation, enabling run-time pruning of the FFT algorithm. The pruning engine allows any pruning pattern without any restrictions on the distribution pattern of the active/inactive sub-carriers, enabling the efficient implementation of dynamic spectrum access (DSA) cognitive radios. The system was prototyped on an ALTERA STRATIX V FPGA in order to evaluate the performance of the pruning engine. Synthesis and simulation results showed that the logic overhead introduced by the pruning block is limited to a 10% of the total resources utilization. Moreover, in presence of a medium-high scattering of the sub-carriers, the power consumption of the FFT core was reduced up to 40%.

1. INTRODUCTION

The continuous evolution of services available through the network has pushed mobile users to increasingly demand more and more bandwidth. Because of the increasing bandwidth demand, police makers and communication technologists had to seek new solutions for resolving the issues of limited spectrum availability. In fact, the spectrum is a limited resource and the spectrum utilization will soon saturate. However, a recent study of the Federal Communication Commission (FCC) has shown that the spectrum scarcity is a false problem, which arises from inefficient spectrum utilization [1]. In fact, from the FCC study it emerges that the spectrum is poorly utilized across frequency, space and time. As a consequence, researchers from industry and academia have developed new paradigms for mitigating the spectrum utilization inefficiency. One proposed paradigm for efficiently utilizing spectrum is dynamic spectrum access (DSA) [13].

To realize DSA communications, highly reconfigurable wireless platforms are needed in order to provide the required spectral agility. Flexible and efficient baseband processing platforms are available through the software-defined radio (SDR) technology and cognitive radio systems [12]. These systems can be utilized to enable DSA communications. Besides the architectural solutions, new agile communication techniques are required to more efficiently exploit the DSA concept: conventional wireless communication systems based for example on frequency division multiplexing (OFDM), might not possess an adequate level of spectral agility, required by DSA communications. As a result, variants of OFDM called non-contiguous OFDM (NC-OFDM) and Discontinuous-OFDM (D-OFDM) were introduced [4] [5].

NC-OFDM systems can transmit data across non-contiguous frequency blocks of sub-carriers turning off the remaining sub-carriers, which are located within the spectral vicinity of existing primary user communications, in order to avoid interference. One of the main advantages of OFDM and its variants is that its implementation of parallel modulated streams of subcarrier data can be efficiently realized using a Fast Fourier Transform (FFT), where each FFT point represents an OFDM sub-carrier. The utilization of NC-OFDM opens opportunities for efficient implementations of the demodulation/modulation blocks. In fact, sub-carriers that are turned off are interpreted by the FFT block as zero-value inputs. Therefore, FFT pruning algorithms could be utilized for lightening the computational load of the FFT block, leading to more power-efficient implementations.

This research work presents the design of an FFT pruning engine, as an add-on block for an existing FFT core. The FFT pruning block allows the elimination of redundant operations at run-time, enabling an efficient implementation of a demodulation block for NC-OFDM systems. The manuscript is organized as follows: Section 2 introduces the theory behind the FFT pruning; Section 3

analyses the state-of-the art in the design and implementation of FFT pruning engines; Section 4 describes in detail the proposed FFT pruning engine and its application to an FFT core; Section 5 and Section 6 present and discuss the achieved results; finally, Section 7 draws conclusions and points out future directions.

2. FFT PRUNING THEORY

FFT pruning was introduced by Markel in [1] as an effective way to reduce the computation complexity of FFT algorithms whenever in presence of zero-valued inputs. The idea behind FFT pruning is to reduce the computational complexity of the algorithm via the elimination of redundant operations, such as: multiplication or addition of neutral terms as well as multiplications by zero factors. In fact, the results of such operations are either a copy of one of the two operands (multiplication/addition by neutral elements) or zero (multiplication by a zero factor) and therefore the mathematical operation can be pruned without any consequences on the correctness of the algorithm. To better underline the simplifications that can be introduced by FFT pruning, Figure 1 presents an 8-point FFT data-flow and an example of input distribution. In the particular example, only 2 out of the 8 inputs are non-zero (x1 and x5). The dashed lines in Figure 1 represent the operations that can be pruned.

The mathematics behind FFT pruning and its advantages have been widely studied. As an example, the computation complexity reduction of DFT and FFT algorithms is discussed in detail in [2] and [3]. From an implementation point of view, the challenges introduced by FFT pruning reside in how to design and implement efficiently the pruning, without any over complications of the control plane. Many different implementations of the pruning algorithm have been proposed.

Alves et al. in [4] present a pruning algorithm based on a configuration matrix and an *if-then-else* statement. The configuration matrix size is *N x log2 N*, where N is the FFT size. Each column of the matrix represents an FFT stage, while the rows represent the input vector for each FFT stages. The matrix is a binary matrix: logical 0 corresponds to a zero-valued input, while logical 1 corresponds to a non-zero input. The computation of the algorithm is then based on a conditional execution of the operations on the basis of the valued stored in the configuration matrix. This approach is based on an extensive utilization of *if-then-else* statements, which potentially limits the advantages of the complexity reduction.

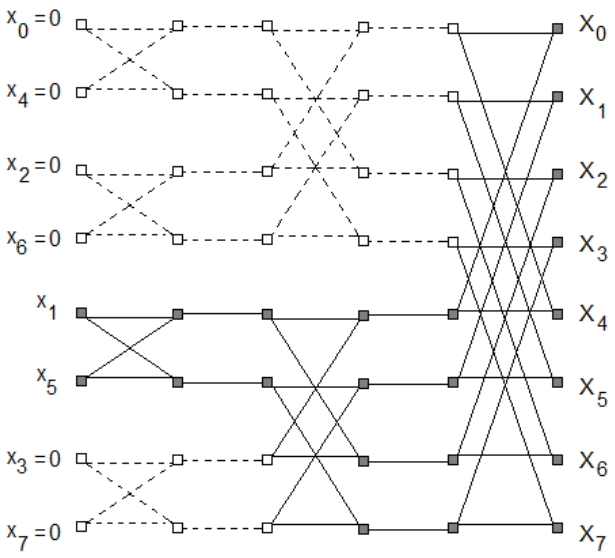

Figure 1: 8-point FFT data-flow and example of pruning simplification.

$$M_{[4]} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad M_{[5]} = \begin{bmatrix} 2 & 4 & 8 \\ 4 & 4 & 0 \\ 5 & 5 & 1 \\ & 6 & 2 \\ & 7 & 3 \\ & & 4 \\ & & 5 \\ & & 6 \\ & & 7 \end{bmatrix} \quad M_{[6]} = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 2 & 0 \\ & 3 & 1 \\ & & 2 \\ & & 3 \end{bmatrix}$$

Figure 2: Configuration matrix for the algorithms proposed in [4][5][6] according to the input distribution presented in Figure 1.

Rajbanshi et al. in [5] introduce a different configuration matrix: Rajbanshi's configuration matrix is composed of *log2 N* columns and *N/2+1* rows. The first row indicates how many non-zero inputs are present for each one of the FFT stages (columns), while the following elements of the column indicate the actual position of the non-zero inputs.

An improved version of Rajbanshi's configuration matrix algorithm is proposed in [6] by Airoldi et al.; the configuration matrix is reduced in size, leading to a more embedded-system friendly implementation. The reduction of the size of the configuration matrix was made possible by storing in the configuration matrix a butterfly identifier instead of the position of each element. However, this

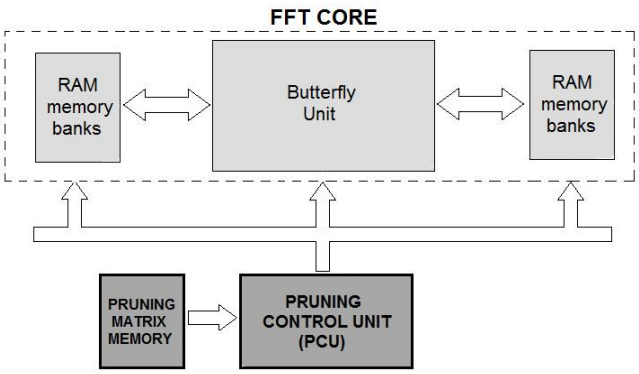

Figure 3: Simplified block diagram of the FFT core and Pruning engine integration.

introduced a restriction in the pruning, allowing the pruning of the butterflies in only the case where both inputs are equal to zero. The effectiveness of the pruning algorithm is then reduced but the actual implementation of the algorithm is simplified, leading to good performance figures. To highlight the differences between the presented algorithms, the configuration matrixes for [4], [5] and [6] are reported in Figure 2. The matrixes were constructed for an 8-point FFT and the example input distribution proposed in Figure 1. From an implementation point of view, the configuration matrixes proposed in [5] and [6] are more suitable for software implementations, while the configuration matrix proposed in [4] can be more efficiently ported to hardware implementations.

3. RELATED WORKS

Different software-hardware solutions have been proposed for the implementation of the FFT pruning algorithm. This section gives an overview of significant research works related to the efficient hardware implementation of FFT pruning for cognitive radios applications.

Venilla et al. in [7] present a 64-point FFT/IFFT with pruning for OFDM-based cognitive radios. The proposed architecture is implemented on a Field Programmable Gate Array (FPGA). To provide the required flexibility for the implementation of the pruning, the work is based on the dynamic partial reconfiguration provided by FPGA technology [10]. The architecture implements a radix-2 FFT algorithm. However, no details are given about the utilization of a configuration matrix for the pruning selection. The results achieved showed a power reduction of 68% when only 16 out of 64 inputs are non-zero. The proposed implementation allows the pruning of the input only for contiguous blocks of 16 inputs at the time, prohibiting the application of the approach to different types of scenario. Moreover, the proposed architecture is tightly bounded to a particular FPGA implementation.

Xu and Lim in [8] propose an FFT pruning design based on a split radix implementation of the FFT kernel. The pruning of the input is managed through an *N x log2 N* configuration matrix. The matrix is a binary matrix and it is similar to the one proposed in [4]. However, the elements of the pruning matrix identify a multiplication and not a single data input. The analysis of the reduction of the computational complexity is evaluated for a 1024-point FFT. Moreover, a hardware implementation of the algorithm (for a 64-point FFT) is presented, but no power consumption and resources utilization figures are given.

Chen et al. in [9] introduce an FFT processor for OFDMA communication systems (e.g. IEEE 802.16e/m and 3gpp-LTE). OFDMA in respect of OFDM allows multi-modal transmission utilizing different numbers of sub-carriers utilization. Therefore, more energy efficient implementation can be achieved through the use of FFT pruning. The proposed FFT architecture is able to perform N-point FFT with N ranging from 128 to 1024. The pruning algorithm introduced relies on OFDMA specifications and therefore is not suitable for cognitive radios applications, where the dynamics of the inputs could be more variegated.

Jang et al. in [15] describe a Synchronous Data Flow (SDF) architecture for the implementation of a 2048-point FFT algorithm. The architecture is described in Verilog HDL and synthetized using Samsung 130nm standard cell libraries. The analysis of power figures underlines a 22.3% reduction of power consumption when the zero input ratio is increased from 30% to 90%.

4. PROPOSED PRUNING ENGINE

The pruning engine proposed by this research work was designed as an add-on block for an existing FFT core [17]. Therefore, the internal control structure of the FFT core was left unaltered. Figure 3 presents a conceptual view of the integration of the FFT core and the pruning engine. Moreover, because the design of the pruning engine was not tailored to the particular FFT core utilized, the proposed solution can be ported to similar types of FFT cores. The design of the pruning engine is composed of two parts: 1) the pruning algorithm and 2) the pruning architecture.

The pruning algorithm defines how the pruning is performed and on which constraints, while the pruning

architecture takes care of the generation of the right control signals for the FFT core, in order to efficiently implement the pruning algorithm.

4.1 Pruning algorithm

The proposed pruning algorithm is based on a pruning matrix. The pruning matrix was derived from [4] and [6]. In
particular, given an algorithm of N-point FFT, the pruning matrix is a $n \times m$ matrix, where n = $log_2(N)$ columns while $m = N/2$. Each column of the matrix represents a stage of the
radix-2 FFT and each one of $N/2$ rows identifies one of the $N/2$ butterflies that compose a given stage (row). The matrix maps the precise location of each butterfly in the data-flow. Each butterfly is represented by one bit (one matrix element), which indicates if the butterfly operation has to be computed or not. To better underline the construction of the pruning matrix, Figure 4(a) presents the pruning matrix for the 8-point FFT example presented in Figure 1. The pruning matrix is stored into a RAM memory. The storage of the pruning matrix is shown in Figure 4(b): a single memory word stores the configuration for up to 32 consecutive butterflies. Finally, the configuration words steer a clock gating system for the activation of the butterfly unit and the memory address generations for the data fetch.

4.2 Pruning architecture

As shown by Figure 3, the FFT core is built around a computational block named Butterfly Unit. The Butterfly Unit can compute two butterfly operations in parallel [14]. A simplified schematic view of one of the two butterfly blocks is given in Figure 5. The figure also highlights the control signals of the pruning engine (*Bfy0_enableX* in the Figure 6). In fact, the pruning control unit (PCU) enables

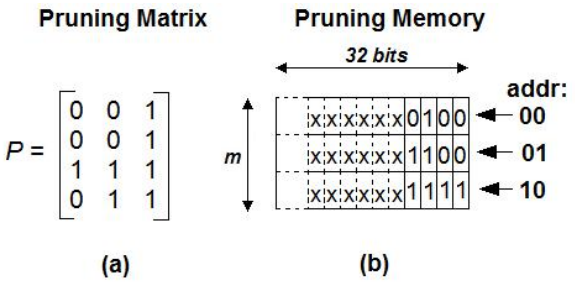

Figure 4: a) Example of pruning matrix in respect of example of Figure 1; b) Memory organization for the storage of the pruning matrix.

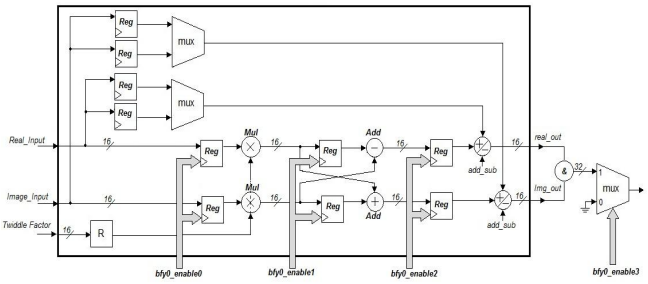

Figure 5: Simplified schematic view of one of the two a butterfly units of the FFT core.

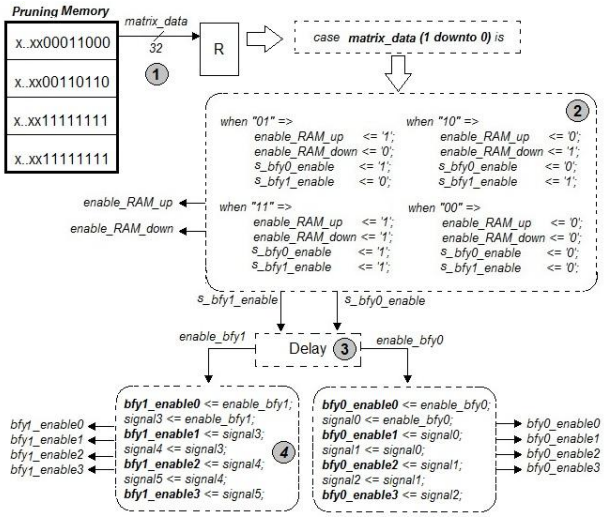

Figure 6: Conceptual view of the Pruning Control Unit design.

or disables the butterfly operations if both inputs are zero. In particular, the enable signals are latched only to the multipliers: the energy consumption of adders is not as significant as the energy consumed by the four multipliers. At each step of the algorithm, the PCU reads and decodes the configurations from the pruning memory and generates the enable signals accordingly. The whole process is timed by the clock of the FFT core in order to keep the two control units synchronized. Indeed, the generation of the enable signals has to respect the pipeline timing of the butterfly unit. Together with the butterfly enables, the PCU generates other two enable signals, which are connected to the two RAM memories. In fact, whenever a butterfly is pruned, the associated reading from the memories is disabled. To better emphasize the steps that characterize the pruning algorithm, Figure 6 provides a simplified view

of the control flow for the PCU. Initially, the PCU fetches from the pruning matrix memory the configurations for the required butterfly operations. The configuration word serves 32 following butterflies within a FFT plane. For larger FFT sizes (requiring more than 32-bit configuration words for each FFT plane) the configuration words are stored at sequential addresses. In such cases, multiple read phases from the memory are needed. In the second phase, the PCU decodes the configuration fetched and generates the enable signals for the data memory accesses. As an example, if the first pairs of bits are both zero, then the calculation of the first two butterflies is avoided. Thus, the enable signals for the memories are not generated. Finally, in the third and last phase, the PCU generates the enable signals for the butterfly units.

FPGA resources	*FFT core*	*Pruning Engine*
Logic Combinational Elements	1057	113
Register	1811	157
DSP blocks	4	0
Memory Size	128 Kbits	11 Kbits

Table 1: Synthesis results of the FFT core and pruning engine on an Altera STRATIX V FPGA.

4.3 Hardware implementation

The pruning engine together with the FFT core were prototyped on a Stratix V GS FPGA (device: 5SGSMD5K2F40C2N) in order to evaluate the overhead of resource utilization introduced by the pruning engine. The synthesis was carried out utilizing Quartus II (release v12.1). Beside area overheads, power measurements were carried out to evaluate the efficiency of the introduced pruning engine. Details about power consumption are given in Section 5. The FFT core is able to perform FFT of different sizes up to 2048-point. Table 1 summarizes the resource utilization for the FFT core and the pruning add-on block. As shown by the resource utilization figures, the pruning engine introduces a logic and memory overhead of about a 10%. In the Table 1 the memory size implemented in the architecture are reported.

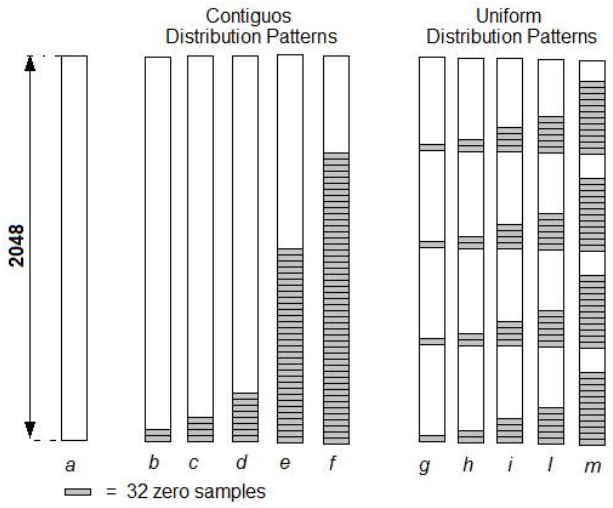

Figure 7: Analyzed input distribution patterns for the evaluation of the FFT pruning engine.

5. RESULTS

The joint system composed of FFT core and pruning engine was evaluated through RTL simulations. In particular, the computation of a 2048-point FFT was considered. Different levels of pruning and different pruning distributions were considered. Figure 7 presents the different input distributions analyzed. The first pattern (pattern *a* of Figure 7) is the reference case where the input sequence contains no zero-inputs. The remaining patterns are divided into two classes: the first considers a scenario where the transmitters delete different contiguous sequences of samples (patterns *b,c,d,e,f*), while the second sub-set shows a uniform distribution of zeros patterns (patterns *g,h,i,l,m*). These distribution topologies represent typical working scenarios in cognitive radio applications, as presented by IEEE 802.22 standard [16]. The estimations of power consumption were performed through the power analyzer tool provided by Altera Quartus II. To obtain more accurate estimations, switching activity files were generated for each one of the analyzed input distribution patterns. Table 1 collects the performance of the pruned FFT architecture in terms of power consumption. Moreover, the Table collects the relative saving obtained via the utilization of the pruning engine in relation to the reference pattern *a*. The reduction of the power consumption mainly depends on two factors: the number of zero samples, and the distribution topology used. For the patterns *b, c, d,* and *f*

increasing the length of the zero sequences leads to higher power savings. Instead, for the patterns belonging at the Uniform Distribution topology, the percentage of the power saved is lower than in the first scenario, because the pruning has a reduced impact on the radix-2 algorithm. In particular, in presence of low pruning levels (e.g. pattern *g*) no power saving is achieved: the advantage of the pruning is compensated by the power consumption of the pruning engine.

Pattern	*Total Power Consumption*	*Power Saved*
a	182 mW	-
b	146 mW	19.8 %
c	144 mW	20.9 %
d	137 mW	24.7 %
e	116 mW	36.3 %
f	106 mW	41.8 %
g	183 mW	-
h	177 mW	2.7 %
i	162 mW	11.0 %
l	138 mW	24.2 %
m	128 mW	29.7 %

Table 2: Estimation of the power consumption of the proposed system for the different operating conditions analyzed.

6. DISCUSSION OF ACHIEVED RESULTS

The results achieved indicate a significant power savings introduced by the pruning engine, leading to a more efficient implementation of the demodulation block in NC-OFDM application scenarios. Moreover, the proposed pruning engine is designed as an add-on block and therefore could be integrated into different FFT cores than the one utilized in this research work. The introduction of the pruning engine does not affect the computation time of the FFT core. On one hand, this design choice limits the power saving achieved (more efficient dynamic power management techniques could be applied), on the other hand it maintains the original specifications of the FFT block, without the need to redesign the interface between the FFT core and the host System on-Chip.

Compared to the power efficiency achieved in [7] the proposed pruning engine offers a reduced power saving. However, the work presented in [7] is strictly tided to FPGA implementations and therefore its application domain might be limited, while the pruning engine proposed in this article is not tided to any particular implementation technology.

In [15] the proposed FFT architecture is not tied to any particular technology either and was synthetized with standard cells libraries. The power consumption is reduced of 22.3% when moving from a 30% to 90% pruning ratio. No power figures are given for an un-pruned scenario. However, the relative power savings achieved by the pruning engine proposed in this article point out a more efficient implementation of the pruning with power saving factors up to 41.8%.

7. CONCLUSIONS

This research work presented an FFT pruning engine as an adds-on block for FFT core architectures. The pruning engine improves the power efficiency of the FFT core without significantly modifying its internal structure and leaving unaltered the FFT core interface. Synthesis and simulation results have shown that for a 10% increase in resource utilization the pruning engine is able to deliver power savings up to 40% in medium high pruning scenarios. Comparisons with related works have shown the competitiveness of the proposed pruning engine.

Future works will cover two directions: 1) ASIC implementation of the proposed pruning engine and 2) a merged implementation of the FFT core and pruning engine, in order to enable the utilization of dynamic power management techniques, such as dynamic frequency and voltage scaling to further improve the power efficiency of the system.

ACKNOLEDGEMENTS

The work was financially supported by the Graduate School in Electronics, Telecommunications and Automation (GETA) and Academy of Finland under contract #258506 (DEFT: Design of a Highly-parallel Heterogeneous MP-SoC Architecture for Future Wireless Technologies). Research grants were received from the Tuula ja Yrjö Neuvo Foundation, the Nokia Foundation, the Ulla Tuominen Foundation and the Tekniikan Edistämissätiö, which are all gratefully acknowledged.

10. REFERENCES

[1] J. Markel, “FFT pruning,” IEEE Transactions on Audio Electroacoustic, vol. 19, no. 4, pp. 305–311, 1971.

[2] H. V. Sorensen and C. S. Burrus, “Efficient computation of the DFT with only a subset of input or output points,” IEEE Transactions on Signal Processing, vol. 41, no. 3, pp. 1184–1200, 1993.

[3] D. Skinner, “Pruning the decimation in-time FFT algorithm,” IEEE Transactions on Acoustic Speech, Signal Processing, vol. 24, no. 2, pp. 193–194, 1976.

[4] R. G. Alves, P. L. Osorio, and M. N. S. Swamy, “General FFT pruning algorithm,” in Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems, vol. 3, 2000, pp. 1192–1195.

[5] R. Rajbanshi, A. M. Wyglinski, and G. J. Minden, “An Efficient Implementation of NC-OFDM Transceivers for Cognitive Radios,” in Proceedings of the 1st International Cognitive Radio Oriented Wireless Networks and Communications Conference, 2006, pp. 1–5.

[6] R. Airoldi, F. Garzia and J. Nurmi, "Efficient FFT Pruning Algorithm for Non-Contiguous OFDM Systems," in Proceedings of the 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP), 2011, pp.144-149.

[7] C. Vennila, C.T.K. Palaniappan, K.V. Krishna, G. Lakshminarayanan, Ko Seok-Bum, "Dynamic partial reconfigurable FFT/IFFT pruning for OFDM based Cognitive radio," in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2012, pp. 33-36.

[8] Yihu Xu; Myong-Seob Lim, "An efficient design of split-radix FFT pruning for OFDM based Cognitive Radio system," in Proceedings of the 2011 International SoC Design Conference (ISOCC), 2011, pp.368-372.

[9] Chao-Ming Chen, Chien-Chang Hung and Yuan-Hao Huang, "An Energy-Efficient Partial FFT Processor for the OFDMA Communication System," IEEE Transactions on Circuits and Systems II: Express Briefs, vol.57, no.2, pp.136-140, 2010.

[10] J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, "Dynamic and Partial FPGA Exploitation," in Proceedings of IEEE, vol.95, no.2, pp.438-452, 2007.

[11] Federal Communication Commission, “Spectrum policy task force report,” ET Docket No. 02135, Tech. Rep., 2002.

[12] J. Mitola, and J.G.Q. Maguire, “Cognitive radio: making software radios more personal,” IEEE Personal Communications, vol. 6, no. 4, pp. 13–18, 1999.

[13] Q. Zhao and B. M. Sadler, “A Survey of Dynamic Spectrum Access,” IEEE Signal Processing Magazine, vol. 24, no. 3, pp. 79–89, 2007.

[14] J. Takala and K. Punkka, “Butterfly Unit supporting Radix-4 and Radix-2 FFT,” in Proceedings of 2005 International TICPS Workshop on Spectral Methods and Multirare Signal Processing (SMMSP), 2005, pp. 47-54.

[15] In-Gul Jang, Zhe-Yan Piao, Ze-Hua Dong, Jin-Gyun Chung and Kang-Yoon Lee, "Low-power FFT design for NC-OFDM in cognitive radio systems," in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp.2449,24.

[16] C. Cordeiro, K. Challapali, D. Birru, and N. Sai Shankar, "IEEE 802.22: the first worldwide wireless standard based on cognitive radios," in Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), 2005, pp.328-337.

[17] D. Revanna, O. Anjum, M. Cucchi, R. Airoldi and J. Nurmi, “A Scalable FFT Processor Architecture for OFDM Based Communication Systems” to appear in Proceedings International Conference on Embedded Computer Systems (SAMOS), 2013.