Reconfigurable FPGA-Based FFT Processor for Cognitive Radio Applications

Mário Lopes Ferreira(✉), Amin Barahimi, and João Canas Ferreira

INESC TEC and Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
{mario.l.ferreira,amin.barahimi,joao.c.ferreira}@inesctec.pt

Abstract. Cognitive Radios (CR) are viewed as a solution for spectrum utilization and management in next generation wireless networks. In order to adapt themselves to the actual communications environment, CR devices require highly flexible baseband processing engines. One of the most relevant operations involved in radio baseband processing is the FFT. This work presents a reconfigurable FFT processor supporting FFT sizes and throughputs required by the most used wireless communication standards. By employing Dynamic Partial Reconfiguration (DPR), the implemented design can adapt the FFT size at run-time and specialize its operation to the immediate communication demands. This translates to hardware savings, enhanced resource usage efficiency and possible power savings. The results obtained for reconfiguration times suggest that DPR techniques are a viable option for designing flexible and adaptable baseband processing components for CR devices.

Keywords: Cognitive radio (CR) · Reconfigurable hardware · Fast Fourier Transform (FFT) · FPGA · Dynamic partial reconfiguration (DPR)

1 Introduction

In the wireless communications technology field, Cognitive Radio (CR) emerged as a solution for a more flexible and efficient spectrum management, enabling dynamic allocation of unoccupied frequency bands and automatic radio settings adjustment for better utilization of the available wireless channels. Regarding baseband processing, CR devices require multi-channel communication methods providing a high degree of flexibility and adaptability. Additionally, the ability to handle high data rates is required. Therefore, a strong candidate for the Physical layer (PHY) baseband processing implementation of CR devices is NC-OFDM (*Non-Contiguous Orthogonal Frequency Division Multiplexing*) [2]. NC-OFDM

M.L. Ferreira—This work was financed by the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within the *CREaTION* project (EXCL/EEI-TEL/0067/2012) and through Ph.D. Grant PD/BD/105860/2014.

V. Bonato et al. (Eds.): ARC 2016, LNCS 9625, pp. 223–232, 2016.
DOI: 10.1007/978-3-319-30481-6_18

is a spectrally agile version of OFDM, a widely used technique which is the base of the most recent wireless communication standards (e.g.: IEEE 802.11, IEEE 802.22, WiMAX, 3GPP-LTE). An indispensable component in OFDM-based systems is the Fast Fourier Transform (FFT) processor. Particularly, in the context of OFDM-based CR applications, the implementation of a flexible and adaptable FFT processor covering the FFT requirements of a wide range of wireless communication protocols is a valuable contribution.

To cover FFT size requirements for several standards, a worst-case approach could be adopted by considering an FFT processor supporting the greatest possible FFT size. However, such a big FFT would not always be needed, leading to an inefficient use of the available resources. Ultimately, this worst-case approach could have a poor behaviour in terms of power consumption. An alternative approach consists of having an FFT processor that is able to adapt its operation and use only the necessary resources for the computational demands at a certain instant, thus improving resource usage and power efficiency. This paper presents work in progress towards the study and design of an efficient, low-power, flexible and reconfigurable FFT processor intended for NC-OFDM transceivers. The FFT processor should support FFT sizes and throughputs required by the most used wireless communication standards and be able to dynamically adapt the FFT size according to the run-time system's operation requirements. Moreover, we explore FPGA-based DPR techniques and evaluate their applicability in the context of CR hardware infrastructures.

2 Related Work

In this section, some FFT implementations for OFDM systems are highlighted. Boopal et al. [3] describe an FFT architecture for variable sizes and multiple streams intended for WiMax wireless receivers. FFT size selection is done using multiplexers. Input data interleaving is used to handle multiple streams. Venilla et al. [11] present a dynamically reconfigurable mixed-radix Single-Delay Feedback (SDF) FFT implementation supporting three FFT lengths (64, 128 and 256 points). The design was implemented on FPGA and the switching between different FFT sizes computation is done by DPR. The DPR implementation showed a better resource usage by saving about 41 % of slices and 32 % of LUTs, compared with a conventional multiplexer-based reconfigurable architecture. No power estimates are presented. Combining DPR with the concept of Time Division Multiplexing (TDM), Chao et al. [4] propose a new FFT design that allows for multiple applications to simultaneously compute variable length FFTs. Wang et al. [12] present a reconfigurable mixed-radix FFT architecture for 3GPP-LTE systems supporting non-powers-of-two FFT sizes. No DPR techniques are employed and run-time reconfiguration is achieved through bypass structures along the pipeline.

Most FFT processors for OFDM systems can be grouped according to whether the supported FFT lengths must be powers of two and according to the reconfiguration methodology employed (DPR versus multiplexing/circuit switching). To our best knowledge, there are no implementations of FFT processors

supporting both powers-of-two and non powers-of-two lengths, and employing DPR.

3 Implementation

3.1 FFT Algorithm

The Cooley-Tukey FFT algorithm [6] was chosen, as it offers a systematic procedure to compute the FFT for any size (N) factorization, with a good balance between arithmetic and computational complexity. This algorithm employs a *divide-and-conquer* strategy to recursively decompose the original N-point FFT into smaller FFTs. Considering an FFT of size $N = N_1 \times N_2$, the Cooley-Tukey algorithm starts by computing N_2 FFTs of size N_1. After that, the outputs of this stage are multiplied by *twiddle factors*. The complex multiplication by these factors is also known as *rotation*. Next, the algorithm proceeds by taking the rotation results and computing N_1 FFTs of size N_2. The most common way to employ the Cooley-Tukey algorithm is to consider values of N which are powers of a base r, such that $N = r^s$. This sub-class of algorithm is commonly referred as *Radix-r FFT* algorithms and the most common radices are 2 and 4, due to their computational simplicity. In a Radix-r FFT algorithm, the basic computational block performs r-points FFTs and is usually called a *butterfly*. Radix-2 algorithms are widely used due to the simplicity of the butterfly. Using butterflies of higher radices will reduce the number of stages and butterflies, but will increase the complexity of the butterfly. However, radix-4 algorithms have some interesting properties, as a considerable number of twiddle factor multiplication consists of multiplying by -j, which is still considered a trivial multiplication. Radices higher that 4 will introduce non-trivial complex multiplications inside the butterflies, increasing the overall algorithm complexity. It is also possible to mix several radices in the same algorithm - *Mixed-Radix FFT* - and thus explore the benefits each radix has to offer.

Typically, FFT sizes are powers-of-two and a Radix-2 Algorithm would cover all required sizes. However, recent standards, as 3GPP-LTE, introduced non powers-of-two FFT sizes (e.g.: 1536-points FFT, required by 3GPP-LTE for 15 MHz bandwidth support). The implemented FFT processor addresses the most frequent FFT sizes in 3G/4G wireless standards. A list of the considered sizes and respective factorization is shown in Table 1. The Radix-4 stages mentioned in Table 1 were implemented using the Radix-2^2 FFT algorithm proposed by He and Torkelson in [7]. This algorithm has the same multiplicative complexity of the Radix-4, while keeping the simple Radix-2 butterfly structures as fundamental processing blocks. Apart from radices 2 and 4, the support for 1536-FFT introduces a Radix-3 stage. Overall, the FFT algorithm variant adopted was the Cooley-Tukey Mixed-Radix-2^2/2/3 algorithm.

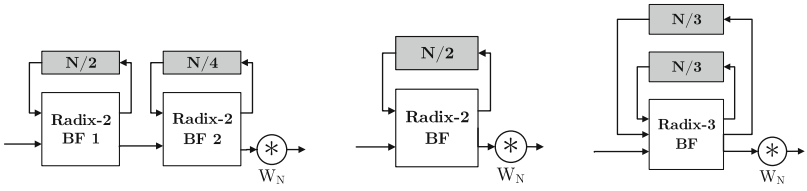
3.2 FFT Architecture

There are two main types of FFT architectures: *Memory-based* and *Pipelined*. Compared with memory-based architectures, pipelined architectures require

Table 1. Implemented FFT sizes and their factorization

FFT size	Factorization	# Radix-4 stages	# Radix-2 stages	# Radix-3 stages
64	4^3	3	-	-
128	$4^3 \times 2$	3	1	-
256	4^4	4	-	-
512	$4^4 \times 2$	4	1	-
1024	4^5	5	-	-
1536	$4^4 \times 2 \times 3$	4	1	1
2048	$4^5 \times 2$	5	1	-

more resources and circuit area, but allow for better performance and the continuous flow of data. In wireless communications, data arrives continuously and FFT requirements are prone to vary with time. So, the FFT processor should have a regular and scalable architecture. Therefore, a pipelined architecture was adopted. Among the existing pipelined architectures, Single-Delay Feedback (SDF) was chosen due to its simpler implementation, lower memory requirements and acceptable throughput. In feedback architectures, the butterflies have feedback loops, so that some butterfly outputs are fed back to a memory stream in the same stage. The memory streams (shift registers) are used to correctly pair the input samples for further butterfly processing. According to the operation phase, an SDF processing element (PE) may forward input data to the feedback memory stream, keeping the butterfly in an idle state, or perform butterfly calculations, producing results to be multiplied by a twiddle factor or fed back to the local memory stream. To implement the Mixed-Radix-$2^2/2/3$ algorithm in a Pipelined SDF architecture, the following basic elements were considered: Radix-2^2 PE, Radix-2 PE and Radix-3 PE. Figure 1 depicts their general structure. The operation parameters in each PE are the size of the feedback memory stream(s) (grey rectangles above the butterflies) and the set of twiddle factors (W_N) used to rotate the butterfly results. Details about the butterfly operations and internal structure can be found in [5,9].



(a) Radix-2^2 PE (b) Radix-2 PE (c) Radix-3 PE

Fig. 1. Basic processing elements

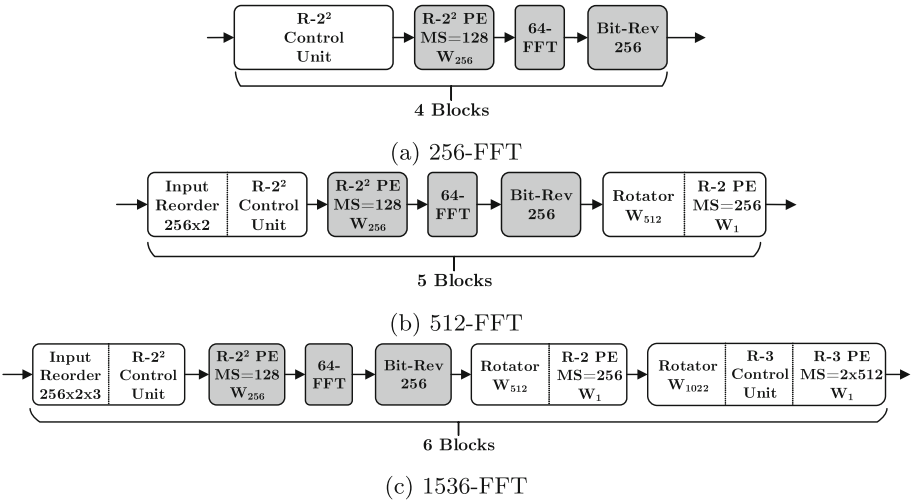


Fig. 2. FFT pipeline structure for N = 256, 512 and 1536

The number of PEs of each type, as well as their location in the pipeline is defined by the factorizations from Table 1 and the chosen FFT algorithm. Figure 2 presents some examples of how to build an FFT pipeline with the PEs previously referred and how to build bigger sizes from smaller ones by reusing some PEs.

For instance, a 256-FFT (Fig. 2a) comprises four Radix-2^2 PEs, three of which belong to the 64-FFT block. The operation of each PE is dictated by the *Radix-2^2 Control Unit*. Apart from feeding input data to the subsequent pipeline stages, this unit generates a binary counter which is propagated along the pipeline. This counter is used by each Radix-2^2 PE to determine whether it should only fed data back to the memory stream, or also perform butterfly computations. Additionally, the binary counter is used to fetch correct twiddle factors for rotation. The considered FFT algorithm receives the inputs in natural order and produces the outputs in bit-reversed order - *Decimation in Frequency*. Thus, in order to provide the outputs in natural order, a bit-reversal operation is required after the last Radix-2^2 PE.

In turn, a 512-FFT (Fig. 2b) can be built by reusing the same Radix-2^2 PEs from the 256-FFT case and adding a Radix-2 PE, whose control is also determined by the binary counter from Radix-2^2 Control Unit. As the factorization of 512 involves different radices, it is necessary to perform a reordering operation on the input data stream and also apply a rotation between the two different radix domains. The relations used in both the input reordering and inter-radix rotations are described in [10]. Similarly, a 1536-FFT (Fig. 2c) is built from the 512-FFT by adapting the reordering operation on the input data and by adding a single Radix-3 stage. Besides the Radix-3 PE and the required rotator between Radix-2 and Radix-3 domains, this later stage includes a *Radix-3 Control Unit*,

as the operations to be performed cannot be correctly controlled by the binary counter from Radix-2^2 Control Unit. The structure and organization of the FFTs for the remaining sizes can be inferred from the examples provided in Fig. 2 and from the factorizations listed in Table 1.

3.3 High-Level Architecture

The reconfigurable Mixed-Radix-$2^2/2/3$ SDF FFT processor was built on a Xilinx ZedBoard platform (FPGA device: XC7Z020-CLG484-1), whose Programmable Logic (PL) section runs at 100 MHz. The arithmetic operations are computed with 16-bit fixed-point numbers for both real and imaginary parts. Both input and output values are in natural order. Figure 3 depicts the high level system architecture of our design.

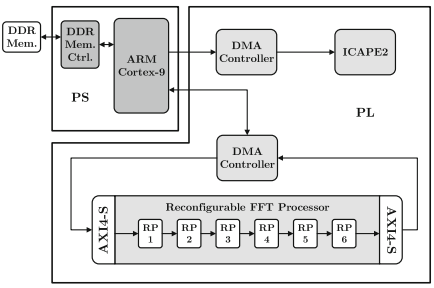


Fig. 3. High level system architecture

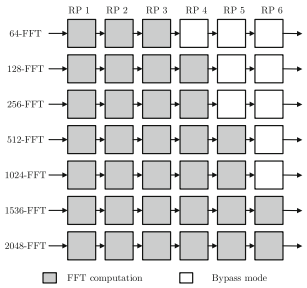


Fig. 4. RPs role for every FFT configuration

The FFT reconfigurable pipeline was implemented on the PL section of the Zynq device. As the bigger FFT sizes (1536 and 2048) are constituted by six building blocks, the pipeline comprises six Reconfigurable Partitions (RPs) embedded in an AXI4-Stream IP core. Depending on the size of the FFT to be computed, a partial bitstream is loaded for every RP using DPR. The FFT processor receives input data from DDR memory, performs computation over these data and sends the results back to DDR memory. Depending on the FFT configuration, RPs are used to perform FFT-related operations or simply propagate results along the pipeline - *bypass mode*. Figure 4 shows the role of each RP for every FFT configuration. Table 2 provides information about the size and available resources for each RP. In order to reduce the amount of data to be transferred though the FPGA configuration port, partial bitstream sizes were reduced by enabling the bitstream compression capability of the Xilinx Vivado tool.

FPGA configuration memory was accessed through the Xilinx Internal Configuration Access Port (ICAPE2) primitive. When powered on, the FPGA is configured with a boot image file stored in an SD card, and partial bitstreams

Table 2. Resources per RP and partial bitstream sizes

	RP1	RP2	RP3	RP4	RP5	RP6
Slice LUTs	400 (0.8 %)	2400 (4.5 %)	1600 (3 %)	1600 (3 %)	800 (1.5 %)	3200 (6 %)
Slice registers	800 (0.8 %)	4800 (4.5 %)	3200 (3 %)	3200 (3 %)	1600 (1.5 %)	6400 (6 %)
BRAM	10 (7.1 %)	10 (7.1 %)	10 (7.1 %)	10 (7.1 %)	10 (7.1 %)	10 (7.1 %)
DSP	0 (0 %)	20 (9.1 %)	20 (9.1 %)	20 (9.1 %)	20 (9.1 %)	20 (9.1 %)
# RM variants	5	3	3	4	3	3
Partial bitstream Max. size (KB)	42.9	156	104	98.3	73	193
Max. Reconfig. time (μs)	112	402	269	254	189	496

are copied from the SD card to the DDR memory. Then, subsequent DPR operations are controlled by the ARM processor. Apart from DDR read/write operations performed by the FFT processor, partial bitstreams need to be fetched from DDR and sent to the ICAPE2 primitive for DPR purposes. These standard bus operations require the CPU to be involved in every transaction. To speed-up both FFT and reconfiguration throughput, CPU involvement in DDR-PL communication was eliminated by using two dedicated DMA controllers for 32-bit data stream transactions: one for data transfers between DDR and ICAPE2 primitive and one for data transfers between DDR and FFT processor. Both DMA controllers were implemented using Xilinx AXI DMA IP cores. In terms of resources, the infrastructure built around the FFT reconfigurable pipeline - *static part* - uses 5410 (10.17 % of available) Sliced LUTs, 6653 (6.25 % of available) Sliced Registers and 6 (4.29 % of available) BRAMs

4 Evaluation and Discussion

The functional correctness of the implemented FFT processor was verified by comparing the system outputs with MATLAB results. For all FFT sizes, the FFT processor produced correct results. To be suitable for OFDM systems, the FFT processor should handle the data rates defined by the most used standards. In steady-state operation, the FFT pipeline throughput is about 98 MSamples/s, which is very close to the limit imposed by the employed FFT architecture at 100 MHz (100 MSamples/s). The achieved throughput is large enough to handle wireless standards as IEEE 802.11a/b/g, WiMAX or 3GPP-LTE.

A CR device should be able to control the adaptability of its PHY operation according to the communication conditions. However, if the reaction to a variation in the communication environment is too slow, interferences with licensed users may occur, thus degrading communication Quality of Service (QoS). So, when designing reconfigurable hardware infrastructures for CR baseband processing, it is important to have a reference regarding the order of magnitude of radio devices reactivity. IEEE 802.22 was the first wireless standard designed for CR environments and it determines several timing parameters to

protect licensed spectrum users from interferences [1]. These timing parameters are a good reference for CR devices reactivity requirements. For instance, radio devices using this standard must be able to establish the communication on a channel in 2 s - *channel setup time* (CST). Then, during normal operation, IEEE 802.22 devices should detect the presence of a licensed user on a channel in less than 2 s - *channel detection time* (CDT). Upon licensed user detection, the device has 2 s to cease all interfering communications in the current channel - *channel move time* (CMT). In particular, events such as transmission establishment/cessation and channel features variation may involve changes in PHY parameters. For our system, the experimental worst-case DPR latency is 1.63 ms, for a reconfiguration throughput of 380 MiB/s (95 % of the ICAP theoretical limit 400 MiB/s). Obviously, NC-OFDM baseband processing does not consist exclusively of FFT computation tasks, but FFT is one of the most demanding tasks within the baseband transceiver.

Compared with the CR timing parameters defined in IEEE 802.22, the worst-case reconfiguration delay observed is within an acceptable range for CR applications. Furthermore, there is still a considerable improvement margin for the work in progress. Reconfiguration time reduction can be achieved by reducing the amount of reconfiguration data to be transferred. Until now, only modular-based DPR was exploited. But, in some situations only small changes are needed in the design and difference-based DPR may be a better option. In this case, only the information regarding differences introduced in the design is contained in the bitstream. If these differences are not large, the bitstream size can be quite small, leading to lower reconfiguration latencies. The way reconfiguration procedures are deployed should be in charge of a Management Unit capable of handling the reconfiguration in an intelligent way, mitigating the impact of reconfiguration on performance and power consumption. As the implemented FFT processor is to be integrated in a reconfigurable NC-OFDM baseband processor, DPR management must be optimized from an overall system perspective.

Resource usage efficiency can be further improved by reusing resources allocated to RPs in bypass mode. For example, instead of computing a single 64-FFT, the three RPs in bypass mode (Fig. 4) could be used to compute another 64-FFT, allowing parallel multiple stream processing and duplication of FFT throughput with the same clock frequency. In the context of an NC-OFDM transceiver, those resources could also be used for other baseband operations (e.g.: constellation (de)mapping, cyclic prefix insertion/removal).

The power consumed by the implementations of FFT processors for the sizes addressed in this work was estimated using Vivado's Power Analysis, and the results also encourage the exploitation of DPR techniques to enhance power efficiency. The estimates are presented in Fig. 5. One can observe that changes in power consumption are mainly due to FFT processor dynamic power and that dynamic power increases with the FFT size. Comparing the smallest and the largest FFT sizes cases (64 and 2048-points FFT), the dynamic power increases by a factor of approximately 3.15. The impact of DPR on power consumption has not yet been evaluated. However, in [8], Liu et al. studied the feasibility

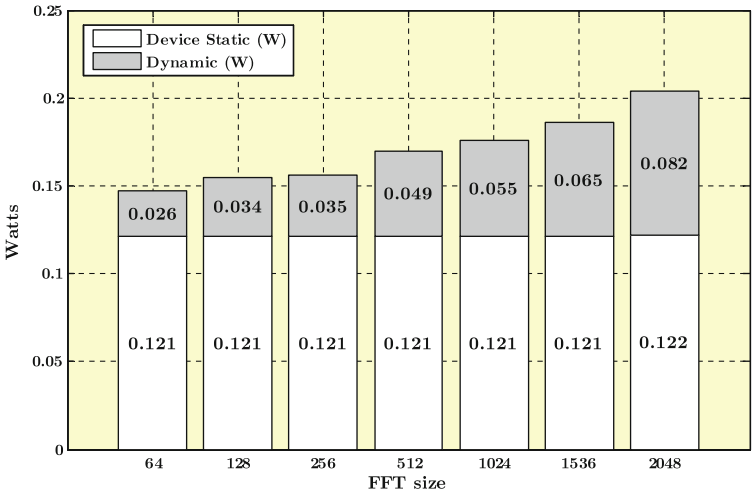


Fig. 5. Power consumption estimations. FPGA device: XC7Z020-CLG484-1; Clock freq.: 100 MHz; Analysis tool: Vivado 2015.2; Post-Implementation power analysis with high confidence level; Activity derived from simulation files (.saif).

of using DPR to improve power consumption and concluded that it can reduce both static and dynamic power. A crucial requirement to mitigate DPR power consumption overhead is the reduction of reconfiguration times.

5 Conclusions

The research work developed so far produced the implementation of a run-time reconfigurable FFT processor supporting FFT sizes and throughputs required by the most used 3G and 4G wireless standards. The FFT processor employs a Mixed-Radix-$2^2/2/3$ SDF approach and run-time reconfiguration was achieved through DPR. The measured reconfiguration times showed the viability of applying DPR techniques in the design of CR baseband processing components. Hence, the implemented FFT processor is suitable for NC-OFDM transceivers in CR applications. For some FFT sizes, there are resources not used for FFT computation purposes which can be reused for other NC-OFDM baseband operations, potentially improving resources usage efficiency. DPR latency introduced in the current design can be further reduced and, although DPR power consumption overhead was not yet studied, estimates for FFT power consumption variation with FFT size motivate the exploitation of DPR for energy-efficient operation.

References

1. IEEE Std 802.22^{TM}-2011: Standard for local and metropolitan area networks - specific requirements - Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Policies and procedures for operation in the TV Bands (2011)
2. Bogucka, H., Kryszkiewicz, P., Kliks, A.: Dynamic spectrum aggregation for future 5G communications. IEEE Commun. Mag. **53**(5), 35–43 (2015)
3. Boopal, P., Garrido, M., Gustafsson, O.: A reconfigurable FFT architecture for variable-length and multi-streaming OFDM standards. In: 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2066–2070 (2013)
4. Chao, H.L., Wu, C.C., Peng, C.Y., Lu, C.H., Shen, J.S., Hsiung, P.A.: Dynamic partially reconfigurable architecture for fast Fourier transform computation. Int. J. Embed. Syst. **6**(2), 207–215 (2014). http://dx.org/10.1504/IJES.2014.063818
5. Cho, I., Patyk, T., Guevorkian, D., Takala, J., Bhattacharyya, S.: Pipelined FFT for wireless communications supporting 128–2048/1536 - point transforms. In: 2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 1242–1245, December 2013
6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**(90), 297–301 (1965). http://www.jstor.org/stable/2003354
7. He, S., Torkelson, M.: A new approach to pipeline FFT processor. In: The 10th International of Proceedings of Parallel Processing Symposium, IPPS 1996, pp. 766–770, April 1996
8. Liu, S., Pittman, R.N., Forin, A.: Energy Reduction with Run-Time Partial Reconfiguration. Tech. rep. MSR-TR-2009-2017, September 2009. http://research.microsoft.com/apps/pubs/default.aspx?id=112466
9. Löfgren, J., Nilsson, P.: On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems. In: NORCHIP 2011, pp. 1–4, November 2011
10. Meyer-Baese, U.: Digital Signal Processing with Field Programmable Gate Arrays, August 2014. http://www.springer.com/engineering/signals/book/978-3-540-72613-5
11. Vennila, C., Lakshminarayanan, G., Ko, S.B.: Dynamic partial reconfigurable FFT for OFDM based communication systems. Circuits Syst. Sign. Process. **31**(3), 1049–1066 (2012). http://link.springer.com/article/10.1007/s00034-011-9367-9
12. Wang, G., Yin, B., Cho, I., Cavallaro, J., Bhattacharyya, S., Takala, J.: Efficient architecture mapping of FFT/IFFT for cognitive radio networks. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3933–3937, May 2014