Cyclostationary Feature Detection on a tiled-SoC

André B.J. Kokkeler, Gerard J.M. Smit, Thijs Krol and Jan Kuper
University of Twente
P.O. Box 217, 7500
AE Enschede, The Netherlands.
Phone: +31 53 4894291, Fax: +31 53 4894571,
a.b.j.kokkeler@utwente.nl

Abstract

In this paper, a two-step methodology is introduced to analyse the mapping of Cyclostationary Feature Detection (CFD) onto a multi-core processing platform. In the first step, the tasks to be executed by each core are determined in a structured way using techniques known from the design of array processors. In the second step, the implementation of tasks on a processing core is analysed. Using this methodology, it is shown that calculating a 127 × 127 Discrete Spectral Correlation Function requires approximately 140 μs on a tiled System on Chip (SoC) with 4 Montium cores.

1 Introduction

Cognitive Radio ([6]) is an emerging area of research where the goal is to find methods to exploit under-utilized electro-magnetic spectrum. Within our AAF project ([1]), Cognitive Radio is proposed for establishing communication infrastructure in emergency situations. This results in two tracks of research conducted in parallel: research on a platform for Cognitive Radio and research on the required cognitive functionality. An essential part within Cognitive Radio is spectrum sensing ([6]). For spectrum sensing, several alternatives have been presented in [7]. The most promising but computationally intensive alternative is Cyclostationary Feature Detection (CFD). CFD allows to exploit the periodicity that especially communication signals exhibit ([2]). This is an important feature for detecting licensed users. In [8], a real time detector for cyclostationary RFI is described. However, the periodicity of the signal to be detected is known, which is generally not the case in Cognitive Radio. No research on the realisation of CFD in the context of Cognitive Radio is known to the authors. In [5], the need for this research is acknowledged and a testbed is proposed, however, without a detailed analysis of mapping CFD onto this testbed.

The digital signal processing requirements of Cognitive Radio heavily depend on the environmental conditions during operation. For that reason, we use the Software Radio concept to guide the research on the platform. To provide both flexibility and power efficiency, we propose to use coarse grain reconfigurable processing elements: the Digital Reconfigurable Baseband Processing Fabric (DRBPF). In this paper, we will elaborate on the mapping of the spectrum sensing part of the Cognitive Radio onto this platform. We introduce a mapping methodology that consists of two steps. In the first step (section 3) we analyse the spectrum sensing part using techniques generally applied to design array processors ([4]). The transformations applied in this step aim at the implementation of spectrum sensing onto an existing reconfigurable platform. For each processing core of the reconfigurable platform, the tasks to be executed and the interconnection structure between cores are determined. In the second step (section 4), the mapping of the tasks onto a processing core is analysed and by means of simulation, an indication of performance is given.

2 Cyclostationary Feature Detection

Cyclostationary Feature Detection (CFD) consists of a combination of an energy detector and a single correlator block. Because we aim at the implementation of CFD in the digital domain, we will give the time discrete expressions for CFD (DCFD). We first define the sampled signal.

$$x_k = x(k \cdot \frac{1}{f_s}) \tag{1}$$

where f_s indicates the sampling frequency. The discrete Fourier Transform is applied to K samples.

$$X_{n,v} = \sum_{k=0}^{K-1} x_{n+k} \cdot e^{j2\pi \frac{n+k}{f_s} v} \tag{2}$$

978-3-9810801-2-4/DATE07 © 2007 EDAA

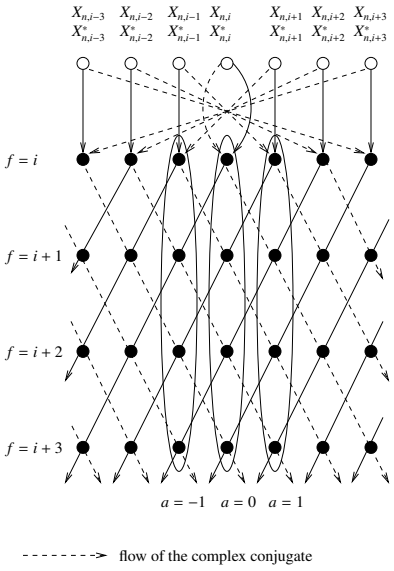

Figure 1. Structure for single n

Finally, the Discrete Spectral Correlation Function (DSCF) is determined.

$$S_f^a = \frac{1}{N} \sum_{n=0}^{N-1} X_{n,f+a} \cdot X^*_{n,f-a} \tag{3}$$

where * indicates the complex conjugate. In case $N = 2^n$, where $n = 1, 2...$, the Discrete Fourier Transform becomes a Fast Fourier Transform (FFT) and the number of complex multiplications that are involved becomes $\frac{1}{2}N\,(^2\log N)$. Determining the DSCF involves $\frac{1}{4}N^2$ complex multiplications. As an example, calculating the DSCF for a 256 point spectrum involves 16 times as many complex multiplications than the determination of the spectrum itself. For the analysis of the platform requirements, we will therefore concentrate on calculating the DSCF.

Determining the DSCF involves a summation over n (expression 3). For each n, a similar type of computation has to be executed which is illustrated in Figure 1. Figure 1 illustrates the structure of the calculations for a single n, $a = -3..3$ and for $f = i, i+1, i+2, i+3$. For other values of a and f, the structure is similar and for that reason in the remainder of this paper we will use examples where $f = 0, 1, 2, 3$ and $a = -3..3$.

The results of the FFT ($X_{n,v}$, expression 2) and their complex conjugates ($X^*_{n,v}$) are at the top of Figure 1. The solid dots represent the multiplications within expression 3. A solid line connects a spectral value $X_{n,v}$ to different multiplications, a dotted line connects a conjugated value $X^*_{n,v}$ to different multiplications. The interconnection pattern connects every multiplication to a 'normal' value and to a conjugated value. Within a row, all S_f^a values are determined for a specific frequency f and within a column, all S_f^a values are determined for a specific frequency offset a. The summation over n is illustrated in Figure 2. For simplicity, we omitted the reshuffling of the conjugated values. For the specific value S_2^{-3}, it is illustrated that it is the result of the summation over n of corresponding multiplications. For all values S_f^a, a similar summation is executed.

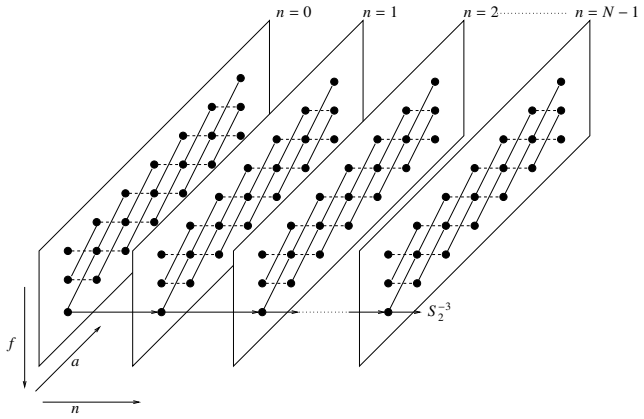

Figure 2. Representation of expression 3

3 Mapping onto a System-on-Chip consisting of reconfigurable tiles (Step 1)

Figure 2 basically presents a three-dimensional Dependence Graph (DG) of DCFD ([4]). Each point of the DG is identified by a vector $\mathbf{v} = (f, a, n)^T$ where ^{T} indicates the transpose operation. Each edge from an 'n-1' plane to the 'n' plane is identified by the 2-tuple $(\mathbf{v}, \mathbf{\Delta v}) = ((f, a, n)^T, (0, 0, 1)^T)$. Traditionally, the DG is used to systematically deduct a multiprocessor architecture. We will follow the traditional approach but the main goal is first to determine the tasks to be executed by the processing cores and second to analyze the interconnection patterns between the processing cores. In the final application we assume that we will use Montium coarse grain reconfigurable processors ([3]) as processing cores. They offer flexibility against acceptable power consumption. Each dot in the DG could represent a (Montium) processor, resulting in an architecture with many processors.

3.1 Tasks to be executed by the processing cores

To reduce the required number of processors, multiple tasks can be mapped onto one processor by means of a processor assignment matrix $\mathbf{P_1}$ and a scheduling vector $\mathbf{s_1}$. The processor assignment matrix $\mathbf{P_1}$ determines on which 'processor' $\mathbf{v}_{new}$ an operation $\mathbf{v}_{old}$ is mapped: $\mathbf{v}_{new} = \mathbf{P_1}^T \mathbf{v}_{old}$. It also determines the displacement parts of the edges: $\mathbf{\Delta v}_{new} = \mathbf{P_1}^T \mathbf{\Delta v}_{old}$. The scheduling vector de-

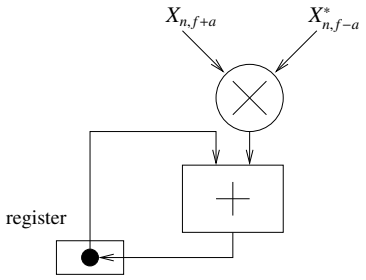

Figure 3. Structure of a processing element after mapping in the n-dimension

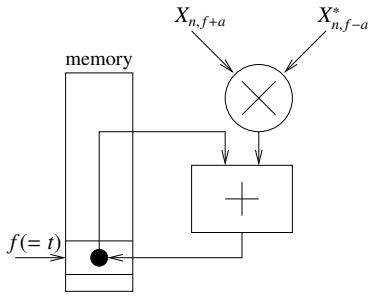

Figure 4. Stucture of a processing element after mapping in the n- and f-dimensions

termines the time at which an operation is executed after mapping: $t = \mathbf{s_1}^T \mathbf{v}_{old}$.

For our application there are numerous possibilities for $\mathbf{P_1}$ and $\mathbf{s_1}$ but we choose a straightforward option:

$$\mathbf{P_1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{s_1} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4}$$

The result is that all operations with identical f and a are executed by the same processor and that the operations in plane $n-1$ are executed before the operations in plane n. The result after mapping is basically a DG equivalent to Figure 1 where each node represents, instead of a complex multiplication, a combination of a complex multiplication and integration (register + adder) as depicted in Figure 3. To further reduce the number of processors, starting with the two dimensional DG in Figure 1, we apply the following processor assignment matrix (which is a vector in the two-dimensional case) and scheduling vector:

$$\mathbf{P_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{s_2} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{5}$$

The result of the processor assignment is a processor array where each processor calculates results for all frequencies f where the results for $f = 0$ are calculated at $t = 0$, etc. Because of this time multiplexing, results of the integration operation have to be stored. The storage location depends on the frequency f (which equals time t). This is depicted in Figure 4.

3.2 Interconnection patterns

Given the processor assignment, the interconnection pattern between the processing elements can be determined. We start by analysing Figure 1 (Note that without limiting the applicability of our approach, we have set $i = 0$). The dotted line originating at the left-most processor for $f = 0$ (which is processed at $t = 0$) indicates that $X^*_{n,3}$ is used by

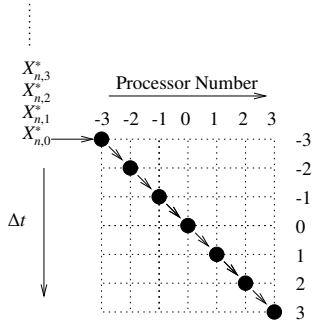

Figure 5. 'Space'-'time delay' diagram

the leftmost processor at $t = 0$, used by the adjacent processor at $t = 1$, and so on. Because of the processor assignment, all dotted lines are mapped on top of each other and therefore share their communication resources. A similar reasoning is valid for the solid lines: they are mapped on top of each other, sharing a second communication structure. Based on this observation, we split the transformation of expression 5.

First, we remove the dependency of absolute time for each set of parallel lines in the DG by means of matrices $\mathbf{P_{2a_1}}$ and $\mathbf{P_{2a_2}}$ where

$$\mathbf{P_{2a_1}} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{P_{2a_2}} = \begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix} \tag{6}$$

For the dotted lines we use matrix $\mathbf{P_{2a_1}}$ to remove absolute time dependence. The result is presented in Figure 5.

For the solid lines we use matrix $\mathbf{P_{2a_2}}$ which results in a Figure similar to Figure 5 but with a flow from top-right to bottom-left.

Second, we do a (trivial) mapping onto a processor array by means of projection matrix $\mathbf{P_{2b}}$ where

$$\mathbf{P_{2b}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{7}$$

Note that the two-stage mapping for determing the interconnection pattern equals the single stage mapping for mapping

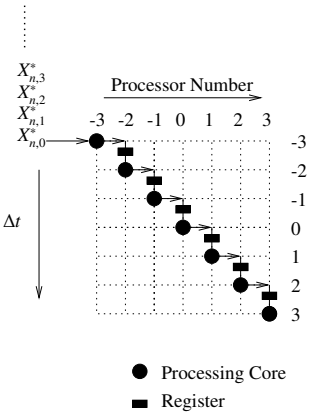

Figure 6. Minimal register structure

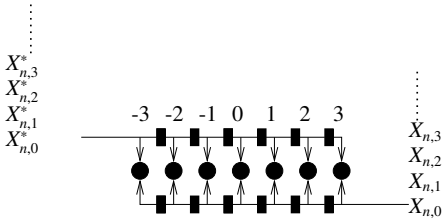

Figure 7. Register based architecture

tasks onto processors by means of $\mathbf{P_2}$ (expression 5) because ${\mathbf{P_{2b}}}^T {\mathbf{P_{2a_1}}}^T = {\mathbf{P_2}}^T$ and ${\mathbf{P_{2b}}}^T {\mathbf{P_{2a_2}}}^T = {\mathbf{P_2}}^T$.

For determining the interconnection pattern, we return to the mapping by matrices $\mathbf{P_{2a_1}}$ and $\mathbf{P_{2a_2}}$. Figure 5 gives the 'space'-'time delay' requirements for the communication path of the conjugated values after mapping by means of $\mathbf{P_{2a_1}}$. There are many options for realizing these communication paths. Below we present a register based solution.

3.3 Register based solution

To obtain the register based solution, we start with the assumption that the propagation delays of data traversing between processors is negligible compared to one period of the clock. Note that this clock also determines the rate at which the multiplications are executed. Consequently, we can only create delays through clocked registers. This means that, to satisfy the requirements in Figure 5, we can only travel in horizontal and vertical directions where we need registers to traverse in the vertical direction. In Figure 6, a communication structure with minimal register usage is given for the conjugated values.

For the non-conjugated values, a similar structure can be drawn from top-right to bottom-left. After the final assignment by $\mathbf{P_{2b}}$ and combining the two communication structures, the result is the architecture presented in Figure 7.

The result is a systolic array where the benefits are: high performance and expandability.

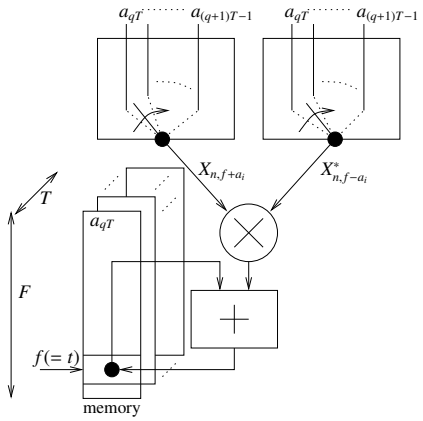

Figure 8. Processing core q

The number of processors (complex multipliers and integrators), however, can still be relatively large. If, for example, 256-point specta are processed, both f and a can range from -63 to $+63$. Consequently, 127 complex multipliers are needed. When realizing CFD on a platform having less than 127 processors, an additional mapping step is required. Again, there are many options of which we will elaborate one.

Suppose the *initial* array of processors ranges from $-(M-1)$ to $M-1$. The total number of initial processors equals $P = 2M+1$. The *final* array consists of Q processors. For a balanced distribution of the load, each final processor has to be able to process at least T tasks from the initial array where

$$T = \left\lceil \frac{P}{Q} \right\rceil \tag{8}$$

and $\lceil\rceil$ indicates the ceil operation. A task p out of the initial array is mapped onto processor q of the final array by means of the following expression.

$$q = \left\lfloor \frac{p}{T} \right\rfloor \tag{9}$$

where $\lfloor\rfloor$ indicates the floor operation. So, the tasks of the intial array mapped onto processor q are tasks qT to $(q+1)T-1$. A resulting processing core after mapping is represented in Figure 8. Both inputs of the multiplier are connected to a switch. These switches are synchronized and select inputs X and conjugated inputs X^* out of the shift registers as indicated in Figure 9, where $T = 4$. The inputs to the switches are stable until T tasks are executed. After that, the elements within the shift registers are shifted one position.

The output of the multiplier is connected to one of the inputs of the adder. The second input of the adder is connected to memory. The memory location selection depends

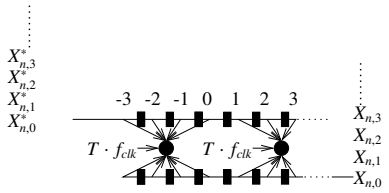

Figure 9. Architecture with multiple tasks on a single processing core

on the frequency f and the tasknumber to be executed. Note that, during the processing of the T tasks in which the inputs to the switch remain constant, the frequency f does not change either. Furthermore, if the total number of frequency points to be processed equals F, the overall memory requirement equals $T \cdot F$ complex values.

4 Mapping onto the AAF DRBPF (Step 2)

Within the AAF project, we plan to map Cyclostationary Feature Detection onto a tiled SoC which consists of four word level reconfigurable processing cores called Montium ([3]). A global overview of a Montium core is given in Figure 10.

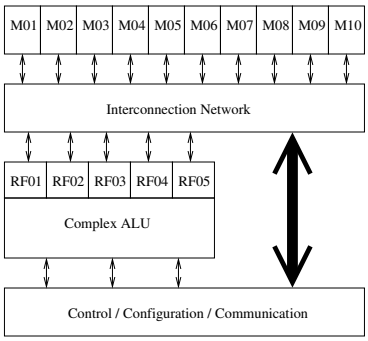

Figure 10. Overview of a Montium core

A Montium core consists of a memory bank of 10 separate memories (M01 to M10) which can be addressed in parallel. The register files of the core (RF01 to RF05) are connected to the memories via an interconnection network. The ALU is tailored towards signal processing applications. It can, for example, execute one complex multiplication per clockcycle. The tasks to be executed by the ALU and the settings of the interconnection network are determined by the control and configuration block which is also used for communication purposes. The processing core as illustrated in Figure 8, has to be mapped onto the Montium core. The mapping is illustrated in Figure 11. The memories for the integration of values are mapped onto Montium memories

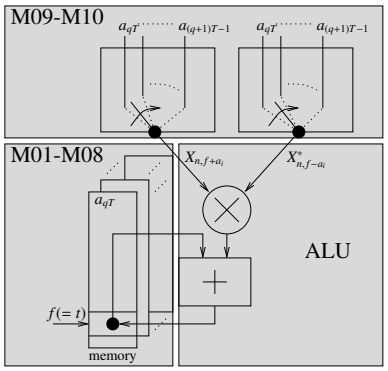

Figure 11. CFD onto a Montium core

M01 to M08. The multiplication and addition are executed by the ALU. The Montium memories M09 and M10 can be used to incorporate those registers of the communication network of which the outputs are connected to the same processing core. The settings of the interconnection network and the control and configuration of the ALU are determined by the control / configuration / communication block and are not illustrated in Figure 11.

Because the memories M09 and M10 are used to realize the registers of the communication structure, data needs to be exchanged between Montium cores. The rate at which data is exchanged is a factor T times lower than the rate at which the basic computation (complex multiplication and integration) is executed. For that reason we assume that the inter-core communication will not significantly influence system performance and the following quantitative analysis will therefore concentrate on the performance of one processing core. Because all processing cores have similar tasks, the results of the performance analysis for one core can be used to determine the performance of the complete multi-core platform.

4.1 Simulation results

The quantitative analysis of mapping CFD onto multiple Montium cores is based on the simulation of an application where 256-point spectra are analysed. Both f and a range from -63 to $+63$ which implies that $P = 127$ and $F = 127$. The platform used within the AAF project consists of 4 Montium cores ($Q = 4$). The number of tasks to be executed by one Montium core is therefore smaller than or equal to 32 ($T = 32$). The number of memory locations needed for storing the results, when accumulating over n, equals $T \cdot F = 32 \cdot 127 < 4K$ complex values or less than 8K real values. The total memory capacity of the Montium memories M01 to M08 equals 8K words of 16 bits. So, for dynamic ranges smaller than 96 dB, the Montium memories are sufficiently large. The communication shift registers

Task	#cycles
multiply accumulate	12192
read data	381
FFT	1040
reshuffling	256
initialisation	127
total	13996

Table 1. Number of processor cycles

are mapped onto memories M09 and M10. Each memory contains 32 complex values. From each memory a value is read every clockcycle. The read-address is generated by an Address Generation Unit (AGU) which accompanies each memory ([3]).

We simulated the tasks for a single Montium tile with the Montium simulator. Table 1 gives an overview of the number of processor cycles required for the different tasks.

The total number of complex multiply accumulate operations equals $T \cdot F = 4064$. Simulations show that a multiply-accumulate requires three clockcycles, so the required number of clock cycles equals 12192. For each 32 multiply accumulate operations, 3 additional clockcycles are needed to read data which leads to 381 additional clockcyles. The DSCF is based on a 256-point spectrum which can be calculated by one Montium in 1040 clockcycles ([3]). The reshuffling of the conjugated values (Figure 1) is done in 256 clockcycles and initially loading the Montium with data requires 127 clockcycles. The total number of clockcycles for 1 integration step in the calculation of a DSCF then equals 13996. The maximum clockspeed of a Montium core equals 100 MHz and therefore the time required for the calculation of one integration step in the calculation of the DSCF equals 139.96 μs.

5 Evaluation of the results

As we stated in the introduction, no comparable studies are known to us. We will therefore limit ourselves to an overview of the most important results.

To analyse 256 samples takes approximately 140 μs. If all samples of a stream are analysed in blocks of 256 samples, an analysed bandwidth of approximately 915 kHz is realised. A single Montium occupies approximately 2 mm^2 using the Philips 0.13 μm CMOS12 process technology. A platform consisting of 4 Montium processors will occupy approximately 8 mm^2. Typical power consumption of a Montium processor is estimated to be 500 μW/MHz. When running on 100 MHz, this results for 4 Montium tiles in 200 mW. In this evaluation we already used the scalability property of the application, the mapping and the platform. The analysed bandwidth, chip area and power consumption scale linearly with the number of Montium processors. This property can be used to estimate performance of other platform configurations.

6 Conclusion

In this paper, Cyclostationary Feature Detection (CFD) is mapped onto a platform consisting of multiple processing cores. A two-step methodology is introduced where in the first step, the functionality is mapped onto the multi-core platform. This results in a set of tasks for each core. In the second step the realisation of these tasks on the processing core is analysed. The advantage of this approach is that the mapping of step 1 is done in a structured way where the communication structure is separated from the processing tasks. Furthermore, the set of tasks for each processing core is almost identical which eases the mapping process in step 2.

For the CFD application, the two-step methodology is elaborated resulting in estimates of performance. A processing platform, consisting of 4 Montium cores is used. The result of the analysis is that on this platform, a spectrum (256 points) and a DSCF (127×127 points) can be determined within approximately 140 μs.

References

[1] The adaptive ad-hoc free band communications (aaf) project websites. 2004-2008. http://www.freeband.nl/project.cfm?id=488.

[2] S. Enserink and D. Cochran. A cyclostationary feature detector. In *28th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 806–810, november 1994.

[3] P. Heysters. *Coarse-Grained Reconfigurable Processors; Flexibility meets Efficiency*. PhD thesis, University of Twente, 2004. ISBN 0-13-942716-3.

[4] S. Kung. *VLSI Array Processors*. Prentice-Hall, 1988. ISBN 0-13-942716-3.

[5] S. Mishra, D. Čabrić, C. Chang, D. Willkomm, B. van Schewick, A. Wolisz, and R. Brodersen. A real time cognitive radio testbed for physical and link layer experiments. In *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN2005*, pages 562–567, november 2005.

[6] J. Mitola. *Cognitive Radio An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, KTH Royal Institute of Technology, Stockholm, 2000.

[7] D. Čabrić, S. Mishra, and R. Brodersen. Implementation issues in spectrum sensing for cognitive radios. In *38th Annual Asilomar Conference on Signals, Systems and Computers*, november 2004.

[8] R. Weber and C. Faye. Real time detector for cyclostationary rfi in radio astronomy. In *EUSIPCO 1998*, pages 1865–1868, september 1998.