

# Cognitive Radio Design on an MPSoC Reconfigurable Platform

Qiwei Zhang · André B. J. Kokkeler ·  
Gerard J. M. Smit

Published online: 13 July 2008  
© The Author(s) 2008

**Abstract** Cognitive Radio has been proposed as a promising technology for solving today's spectrum scarcity problem by means of dynamic spectrum access. The multiprocessor system-on-chip (MPSoC) reconfigurable platform is proposed as an enabling technology for cognitive radio. In this paper, we propose a design methodology based on task transaction level interface for the design of cognitive radio baseband on an MPSoC reconfigurable platform. The reconfiguration of a novel, low-complexity fast Fourier transform for orthogonal frequency-division multiplexing based Cognitive Radio is used as a design case to show the effectiveness of the methodology for modelling the dynamic behavior of Cognitive Radio and facilitating the platform implementation.

**Keywords** cognitive radio · MPSoC · OFDM · design method · task transaction level interface · sparse FFT

## 1 Introduction

Recent studies have shown that most of the assigned radio spectrum is underutilized. On the other hand, the increasing number of wireless multimedia applications leads to increasing spectrum scarcity. Cognitive

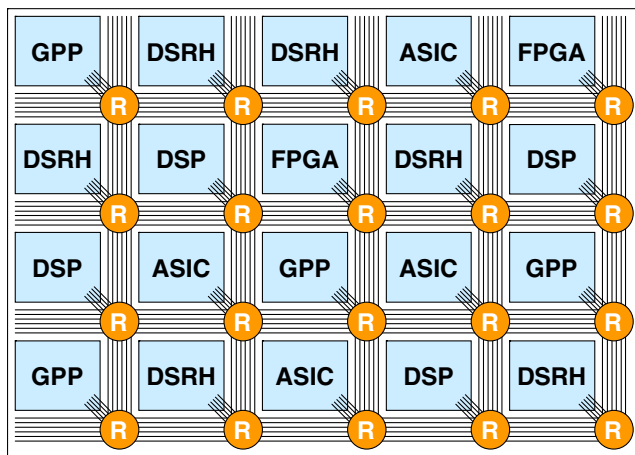
Radio [1, 2] is proposed as a promising technology to address the paradox of spectrum scarcity and spectrum under-utilization. In Cognitive Radio, spectrum sensing locates the unused spectrum segments in a targeted spectrum pool. These segments will be used optimally without harmful interference to licensed users. This technology is called *spectrum pooling* [3]. In spectrum pooling, orthogonal frequency-division multiplexing (OFDM) is proposed as the baseband transmission scheme. Those subcarriers which cause interference to licensed users should be nullified. Therefore, OFDM based Cognitive Radio has to be reconfigurable to be adaptive to the changing wireless channels. This reconfigurability has to be supported by a reconfigurable platform. Our research undertaken in the Adaptive Ad-hoc Freeband (AAF) project ([www.freeband.nl](http://www.freeband.nl)) focuses on mapping the baseband algorithms of Cognitive Radio onto a reconfigurable platform.

## 2 Multiprocessor system-on-chip for cognitive radio

Cognitive Radio is seen as an evolution from the software-defined radio platform [1]. However, the traditional software-defined radio platform for digital processing is mainly based on General Purpose Processors and Digital Signal Processors which are inadequate for future high data rate wireless communications in terms of processing speed and energy efficiency. With the advance of the semiconductor technology, the future trend of wireless baseband processors is moving toward Multiprocessor System-on-Chips (MPSoCs) which integrate heterogeneous processing elements tailored for different processing tasks. MPSoCs offer high performance, reconfigurability and energy efficiency.

---

Q. Zhang (✉) · A. B. J. Kokkeler · G. J. M. Smit  
Department of Electrical Engineering, Mathematics  
and Computer Science, University of Twente,  
Enschede, The Netherlands  
e-mail: q.zhang@utwente.nl



**Figure 1** Heterogeneous multiprocessor tiled SoC

Therefore, we proposed a tiled MPSoC (see Fig. 1) architecture to support Cognitive Radio [4]. These tiles can be various processing elements including General Purpose Processors, Field Programmable Gate Arrays, Application Specific Integrated Circuits and Domain Specific Reconfigurable Hardware (DSRH) modules which target the specific algorithm domains. The Montium [5] tile processor developed at the University of Twente is an example of a DSRH. It targets the digital signal processing algorithm domain and has the flexibility to adapt to different algorithms in an energy-efficient manner. Therefore, the Montium tiled processor is the key element in our proposed reconfigurable platform for Cognitive Radio. The tiles in the SoC are interconnected by a Network-on-Chip (NoC). Both the SoC and NoC are dynamically reconfigurable, which means that the programs (running on the reconfigurable processing elements) as well as the communication links between the processing elements are configured at run-time.

### 3 TTL design methodology

MPSoCs offer many advantages as described in the previous section. However, it is a challenging task to map applications onto MPSoCs. First, the applications to be mapped on the MPSoC become more complex: they consist of more and more tasks and some of the tasks may change their behavior dynamically. Second, in order to map tasks to different components on an MPSoC, designers have to deal with the low-level interfaces for the inter-component communication and synchronization which become a bottleneck from a performance and an energy point of view. Further, opportunities for the reuse of hardware and software

modules are limited and no method exists for exploring their trade-offs. Therefore, there is a gap between the application models used for specification and the optimized implementation of the application on an MPSoC. A task transaction level (TTL) interface approach [6] was proposed to help to close the gap by raising the abstraction level. We propose to use the TTL approach both for developing the Cognitive Radio application at the system level and as a platform interface for implementing the application onto the MPSoC architecture.

In the TTL approach, an application is modelled as a task graph. A task is an entity that performs computations. One task may communicate with other tasks via channels. Communications are invoked by calling TTL interface functions. Computation components can be plugged-in and replaced as functions, which allows exploring and validating design alternatives at a high level of abstraction. The TTL model of the application can be implemented on the targeted platform providing that the TTL shells are available for each processor. A design example on an OFDM receiver for HiperLAN/2 was given in [7]. The TTL approach is extended for modelling reconfigurable applications such as adaptive baseband processing in Cognitive Radio.

### 4 Design case: a low complexity fast Fourier transform for OFDM based cognitive radio

In this paper, we present a design case of a reconfigurable low complexity fast Fourier transform (FFT) algorithm for OFDM based Cognitive Radio to demonstrate the TTL approach. The reasons to choose this design case are: 1) The FFT is an essential and the most computational intensive task in OFDM baseband processing; 2) the algorithm is novel in the context of Cognitive Radio; 3) the algorithm is reconfigurable. We organize this section as follows. First, OFDM based Cognitive Radio is introduced. Then our proposed low-complexity FFT is explained and followed by the TTL implementation. Finally the results will be analyzed.

#### 4.1 OFDM based cognitive radio

Theoretically, an OFDM-based Cognitive Radio system can optimally approach the Shannon capacity in the segmented spectrum by adaptive resource allocation on each subcarrier, which includes adaptive bit loading and adaptive power loading. In [4], we proposed an OFDM system with adaptive bit loading and power loading for Cognitive Radio. We could maximize

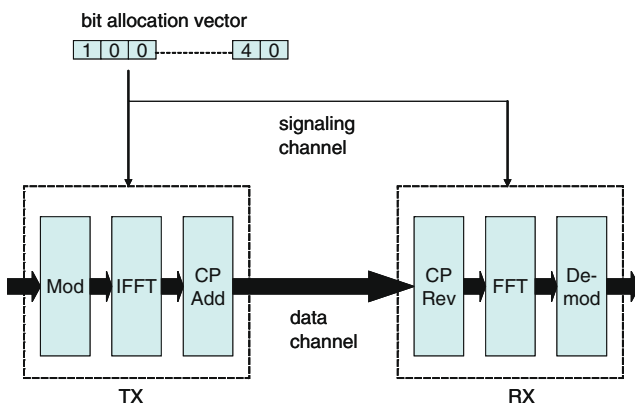
the data rate of the system under a certain power constraint. It is formulated as follows:

$$\begin{aligned} \text{Max } R &= \sum_{k=1}^K \frac{F_k}{K} \log_2 \left( 1 + \frac{h_k^2 p_k}{N_0 \frac{B}{K}} \right) \\ \text{Subject to: } \sum_{k=1}^K p_k &\leq P_{\text{total}} \\ F_k &\in \{0, 1\} \text{ for all } k \\ p_k &= 0 \text{ for all } k \text{ which satisfies } F_k = 0 \end{aligned} \quad (1)$$

where  $R$  is the data rate;  $K$  is the number of the subcarriers;  $N_0$  is the noise power density,  $B$  is the band of interest for Cognitive Radio,  $h_k$  is the subcarrier gain and  $p_k$  is the power allocated to the corresponding subcarrier.  $F_k$  is the factor indicating the availability of subcarrier  $k$  to Cognitive Radio, where  $F_k = 1$  means the  $k$ th carrier can be used by Cognitive Radio. The system power minimization can also be applied under the constraint of a constant data rate. We formulate it as follows:

$$\begin{aligned} \text{Min } \sum_{k=1}^K p_k &= P_{\text{total}} \\ \text{Subject to: } R &= \sum_{k=1}^K \frac{F_k}{K} \log_2 \left( 1 + \frac{h_k^2 p_k}{N_0 \frac{B}{K}} \right) \\ F_k &\in \{0, 1\} \text{ for all } k \\ p_k &= 0 \text{ for all } k \text{ which satisfies } F_k = 0 \end{aligned} \quad (2)$$

A functional diagram of the system is presented in Fig. 2. A bit allocation vector indicates how many bits are loaded on each subcarrier. The number of bits corresponds to the different modulation types used for each subcarrier. The bit allocation vector is determined by the spectrum occupancy information from spectrum

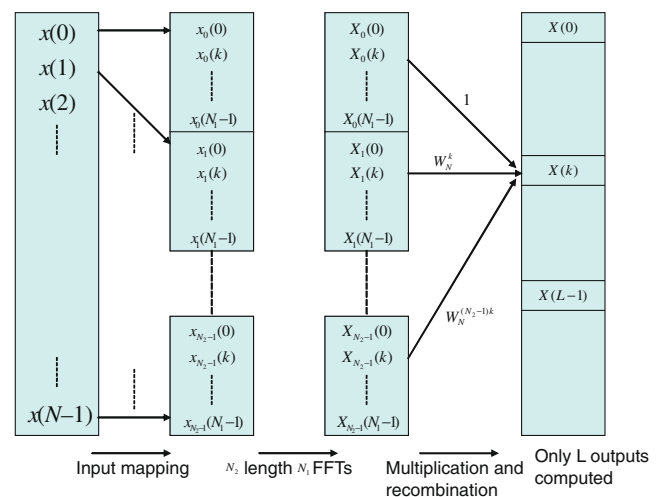


**Figure 2** OFDM for cognitive radio

sensing and the SNR of subchannels. The bit allocation vector is disseminated via a signaling channel, so that both transmitter and receiver have the same information. We assume the bit allocation vector does not change frequently for instance during several frames. The basic idea is to load more bits on good subcarriers and load zeros onto carriers which cause interference to the licensed user or lead to poor transmissions. The expectation is that there could be a large number of zero inputs/outputs for the IFFT/FFT when a large part of the spectrum is not available to Cognitive Radio or there are many bad channels. Therefore the normal radix-2 IFFT/FFT will be inefficient in this case due to the wasted operations on zeros.

#### 4.2 Sparse FFT

In mathematical terms, arrays or matrices where most of the elements have the same value (called the default value, usually 0) and only a few elements have a non-default value are *sparse*. It is beneficial and often necessary to take advantage of the sparse structure algorithmically to reduce the operations of the standard algorithms. Therefore, we propose a low complexity FFT algorithm as an option for OFDM based Cognitive Radio in [8]. We term this FFT algorithm *sparse FFT*. The algorithm is based on transform decomposition in [9], but has been tailored for our Cognitive Radio system. Transform decomposition can be seen as a modified Cooley–Tukey FFT where the DFT is decomposed into two smaller DFTs. The detailed mathematical derivation of the algorithm can be found in [9]; here we only show the computational structure in Fig. 3. We made some modifications to the original algorithm



**Figure 3** Computational structure of transform decomposition

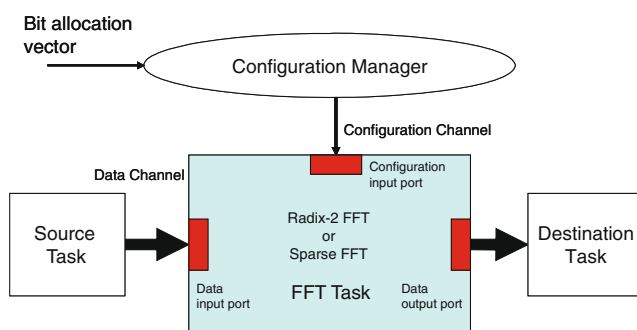
to facilitate efficient hardware implementations. We choose the total number of carrier  $N$  as a power-of-two integer and  $L$  is the number of non-zero outputs.  $N_1$  is chosen as a nearest power-of-two integer larger than  $L$ . This choice of  $N_1$  helps to exploit more regularities. Thus  $N_2$  is also a power-of-two integer which satisfies  $N = N_1 N_2$ . The algorithm is decomposed into two major parts: the  $N_2$  blocks of  $N_1$ -point DFTs which can be implemented as radix-2 FFTs and the multiplications with twiddle factors and the recombination of the multiplications. The reduction of computation comes from the second part where only  $L$  twiddle factors are multiplied with each  $X_{n_2}(\langle k \rangle_{N_1})$  (denoting modulo  $N_1$ ) for  $n_2 = 1, 2, \dots, N_2$ . According to the computational structure, we perform quantitative analysis on the computational complexity by counting the number of complex multiplications. So the number of multiplication for sparse FFT is  $(N_2 - 1) * L + \frac{N}{2} \log_2 N_1$ , which is less than the number of multiplications for radix-2 FFT ( $\frac{N}{2} \log_2 N$ ) when  $L < N/2$ .

Therefore, we propose that the system will be reconfigured to a sparse FFT when there are a large number of zeros in the bit allocation vector.

#### 4.3 The TTL implementation

We implemented the reconfigurable sparse FFT in the TTL environment to achieve the following goals: 1) to verify the sparse FFT at system level; 2) to obtain high level profile information in terms of computation workload and communication workload which help to make the implementation trade-offs on processors.

During the first step, we create the task graph (see Fig. 4) of the reconfigurable sparse FFT. The source task generates the input samples which are sent via the data channel to the FFT task. The destination task consumes the output samples from the FFT task. A configuration manager decides the type of FFT algorithm, depending on the number of non-zero values



**Figure 4** Task graph of reconfigurable sparse FFT

$L$  in the bit allocation vector. If  $L < N/2$ , the configuration manager will generate the configuration data for a sparse FFT. Then it indicates the FFT task to perform sparse FFT and sends all the configuration data to the FFT task via the configuration channel. The configuration manager will go to standby until a new bit allocation vector arrives. Depending on  $L$ , the FFT task either performs radix-2 FFT or the sparse FFT.

The TTL functions are called from the TTL C/C++ library to create tasks, define communication interfaces and generate the task graph. At the system level, the tasks are coded in C/C++. But in the platform implementation, the tasks can be implementations on a particular processor. Here we give a pseudo code example of the TTL implementation to show how the reconfiguration is done for the FFT task.

```

Task Task_FFT
{initialization;
while(true)
{local variables;
\\check the configuration updates
tryAcquiredata(Task_FFT->config_inport)
{\\update L
ttl_read(Task_FFT->config_inport, L);
\\read in configuration
ttl_read(Task_FFT->config_inport,
SFFT_CONFIG_DATA);
}
\\read in data
for(i=0; i<num_samples; i++)
ttl_read(Task_FFT->data_inport,
proc_buffer[i]);
\\sparse FFT or radix-2 FFT
if (L<num_samples/2)
{\\sparse_FFT processing
call sparse_FFT;
}
else
{\\radix-2 FFT
call rad2_FFT;
}
\\write out results
for(i=0; i<num_samples; i++)
ttl_write(Task_FFT->data_outport,
proc_buffer[i]);
}
}

```

The FFT task checks the updates from the configuration channel. If a new configuration is generated by the configuration manager, the FFT task will read in the configuration data from the configuration channel via the configuration input port. Then the FFT task

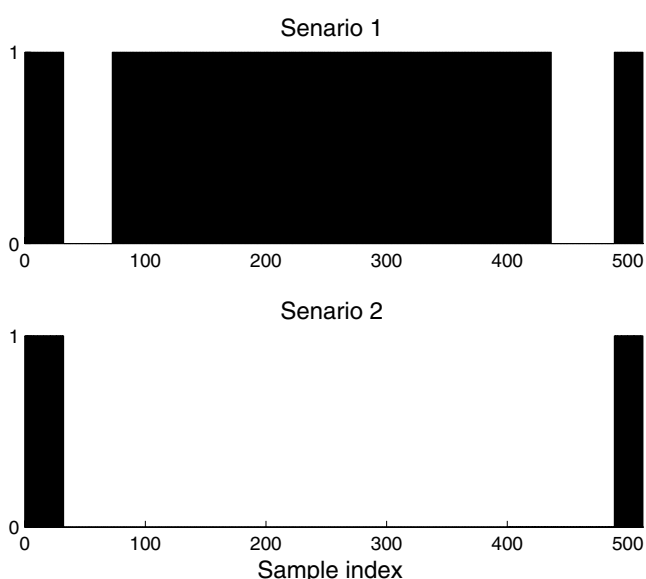
**Table 1** OFDM parameters: sample frequency and symbol duration

$B = f_s$ [MHz]	N	$\Delta f$ [kHz]	$T_u$ [ $\mu s$ ]
5.12	512	10	100

reads in samples from the source task. After reading the samples and the configuration data, the sparse FFT or radix-2 FFT procedure will be executed depending on  $L$ . After the FFT processing, the results will be written out to the data channel via the output data port. Both synchronization and data transfer are done by the TTL read and write functions. The procedures for sparse FFT and radix-2 FFT are software implementations in C/C++, but they can also be replaced by equivalent hardware implementations; for example, configurations for the Montium processor.

#### 4.4 Results

We applied the reconfigurable sparse FFT to an OFDM receiver based on the specification in [10] for the AAF system. The OFDM parameter set under consideration is shown in Table 1. The bandwidth for the OFDM system is 5.12 MHz and there are 512 subcarriers in one OFDM symbol. Thus, subcarrier spacing  $\Delta f$  is 10 kHz and the useful symbol part duration  $T_u$  is 100  $\mu s$ . Therefore, we need a 512-point FFT to process an OFDM symbol, where some subcarriers might be zero according to the bit allocation vector. In Fig. 5, we show an example of the sparse FFT reconfiguration for the given OFDM system. We denote all the zero output indexes of the FFT with 0 and non-zero indexes

**Figure 5** Example of reconfiguration**Table 2** Computation workload of the FFT task

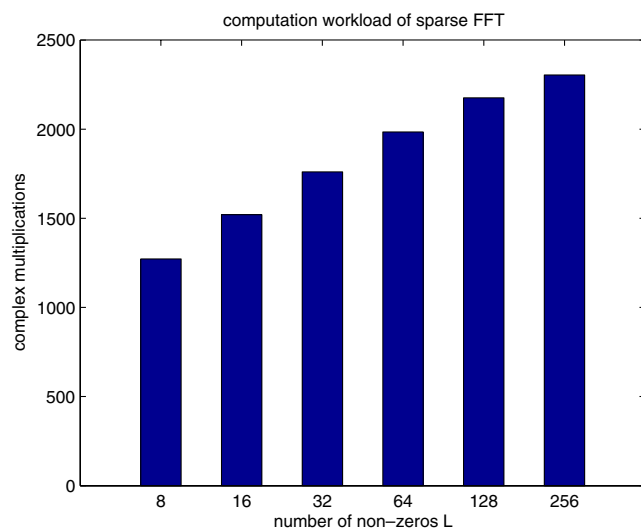
	Complex multiplication (instruction count)
$S_1$	2304
$S_2$	1928

with 1. In Scenario 1, 420 of 512 indexes are non-zeros which means that most of the subcarriers can be used by Cognitive Radio. To avoid causing interference to a potential licensed user, Cognitive Radio has to switch off a certain number subcarriers and re-assign the transmitted information to the available subchannels. This corresponds to Scenario 2 where only 56 out of 512 indexes are non-zeros. From Scenario 1 to Scenario 2, the FFT task is reconfigured from a radix-2 FFT to a sparse FFT. The high level TTL implementation has been run on a Linux PC. The computation result verifies the functional correctness of the sparse FFT. The TTL run-time environment can generate high level profile information in terms of computation workload and communication workload. The computation workload is measured by counting the number of annotated instructions while the communication workload is measured by counting the number tokens (data units) that are travelling through the TTL channels. Table 2 shows the computation workload of the FFT task in two scenarios generated by TTL. The reduction of computation in Scenario 2 is due to the sparse FFT which takes advantage of sparse data. Considering that the worst case execution time for processing an OFDM symbol is 100  $\mu s$ , we estimate the minimum required processing capacity for the  $S_1$  and  $S_2$  in Table 3. The profile information generated by the TTL run-time environment is independent of platforms. However, it can help to generate the platform dependent profile for specific implementations. By associating execution times with instructions, and by multiplying these execution times with the instruction counts, one can obtain a rough estimate of total execution time of a task on a certain processor. Considering the Montium for the FFT task, the Montium can execute one complex multiplication instruction in one clock cycle. From Table 3, we find that the Montium has to run at at least 23 MHz for Scenario 1 and 19 MHz for Scenario 2. In other words, the sparse FFT will save 16% processing capacity. Such a reconfiguration only takes place when the bit allo-

**Table 3** Minimum processing requirements

	Complex multiplications (per second)
$S_1$	$23 \times 10^6$
$S_2$	$19 \times 10^6$





**Figure 6** Computation workload of sparse FFT for 512 samples

cation vector has been updated. We expect the bit allocation vector not to change very often: at least it will be constant over several OFDM frames. Therefore the reconfiguration overhead is relatively small compared to the saving of computations. From the TTL profile information, we compare the computation workload of 512-point sparse FFT for various  $L$  with different zero distributions. Figure 6 shows that the computation workload increases with the number of non-zero  $L$ .

## 5 Future work

Based on the TTL model, the reconfigurable sparse FFT will be mapped onto the proposed MPSoC platform. Some software implementations are ready to be ported to the platform. For example, the C code for the configuration manager can be compiled for an ARM processor. Hardware implementations such as the Montium configuration for the sparse FFT will also be ported to the platform. Our ultimate goal is to develop the entire Cognitive Radio baseband application in the TTL framework. Based on the TTL model, the Cognitive Radio baseband application will be mapped onto the proposed MPSoC platform.

## 6 Conclusion

In this paper we have presented a system level design methodology for mapping Cognitive Radio onto an MPSoC platform. The design methodology is based on a TTL interface to partition the application into communicating tasks. By making the communication explicit, the computation (task) is implemented sepa-

ately from the communication. We show a design case on a novel FFT for OFDM based Cognitive Radio. This sparse FFT takes advantage of sparse data and results in a computation complexity reduction. The reconfigurable sparse FFT has been implemented in the TTL framework. From the design case, we conclude that TTL offers the following advantages:

- the functional correctness of the algorithm has been verified at a system level
- the TTL programming model facilitates the application partitioning and the task based implementation on MPSoCs
- the profile information given by the TTL run-time environment is used for the computation and communication complexity analysis at a system level; therefore design trade-offs can be made at an early design stage
- TTL is suitable for modelling a reconfigurable application
- the TTL code is portable to the reconfigurable platform.

In conclusion, TTL is a suitable design methodology for mapping Cognitive Radio on a reconfigurable MPSoC. It brings Cognitive Radio from a novel idea closer to a reality.

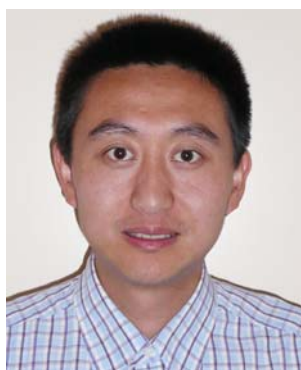
**Acknowledgements** We acknowledge the support for the TTL interface approach by Philips Research. The work is sponsored by the Dutch Ministry of Economic affairs Freeband AAF project. The authors would like to thank their colleagues from the International Research Centre for Telecommunications-transmission and radar of the Technical University Delft and the Signal and System group at the University of Twente for the fruitful discussions in the AAF project.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

## References

1. Mitola J III (2000) Cognitive radio: an integrated agent architecture for software defined radio. PhD thesis, Royal Institute of Technology
2. Haykin S (2005) Cognitive radio: brain-empowered wireless communication. *IEEE J Sel Areas Commun* 23(2):201–220, February
3. Weiss TA, Jondral FK (2004) Spectrum pooling: an innovative strategy for the enhancement of spectrum efficiency. *IEEE Commun Mag* 42:S8–S14
4. Zhang Q, Kokkeler ABJ, Smit GJM (2006) A reconfigurable radio architecture for cognitive radio in emergency networks. In: European conference on wireless technology, Manchester, September

5. Heysters P (2004) Coarse-grained reconfigurable processors; flexibility meets efficiency. PhD thesis, University of Twente
6. van der Wolf P, et al. (2004) Design and programming of embedded multiprocessors: an interface-centric approach. In: Proceedings of ISSS+CODES, Stockholm, 8–10 September 2004
7. Zhang Q, Kokkeler ABJ, Smit GJM (2006) Adaptive OFDM system design for cognitive radio. In: 11th international OFDM-workshop 2006, Hamburg, August 2006
8. Zhang Q, Kokkeler ABJ, Smit GJM (2007) An efficient fft for OFDM based cognitive radio on a reconfigurable architecture. In: IEEE CogNet 2007 Workshop, Glasgow, 24–28 June 2007
9. Sorensen HV, Burrus S (1993) Efficient computation of the DFT with only a subset of input or output points. IEEE Trans Signal Process 41:1184–1200
10. Hoeksema FW, Heskamp M, Schiphorst R, Slump CH (2005) A node architecture for disaster relief networking. In: Proceedings of the first IEEE symposium on new frontiers in dynamic spectrum access networks (DySPAN2005), Baltimore, 8–11 November 2005



**Qiwei Zhang** received his B.S. degree in electrical engineering from Jilin University, Changchun, China, in 2002 and his M.Sc. (with distinction) degree in wireless communication from the University of Southampton, Southampton, UK, in 2004.

Currently he is working towards his Ph.D. degree in the University of Twente, Enschede, The Netherlands. His research interests include Cognitive Radio, wireless communication, signal processing and low-power embedded system design.



**André B.J. Kokkeler** received his M.Sc. degree in electrical engineering from the University of Twente, Enschede, The

Netherlands. He finished his Ph.D. thesis entitled “Analog-Digital Codesign using Coarse Quantization” in 2005. He currently is an assistant professor with the faculty of EEMCS, University of Twente, where he is involved in research projects, sponsored by the Dutch government and industry.

After receiving the M.Sc. degree, he worked for eight years at the Netherlands foundation for research in astronomy (ASTRON) as a scientific project manager and more than 6 years at Ericsson as a system engineer. Since 2003 he works at the University of Twente. He has a background in telecommunication, mixed-signal design and signal processing (beamforming). Currently, his main interest lies in the area of applying low-power design techniques for computationally intensive applications. The emphasis is on reconfigurable architectures for streaming applications.



**Gerard J.M. Smit** received his M.Sc. degree in electrical engineering from the University of Twente, Enschede, The Netherlands. He finished his Ph.D. thesis entitled “The Design of Central Switch Communication Systems for Multimedia Applications” in 1994. He currently is a Full Professor with the faculty of EEMCS, University of Twente, where he is responsible for a number of research projects sponsored by the EC, industry, and Dutch government in the field of multimedia and efficient reconfigurable systems.

After receiving the M.Sc. degree, he worked for four years at the Research Laboratory of Océ, Venlo, The Netherlands. In 1994, he was a Visiting Researcher with the Computer Laboratory, Cambridge University, Cambridge, MA, and, in 1998, he was a Visiting Researcher with Lucent Technologies Bell Labs Innovations, Murray Hill, NJ. Since 1999, he has been leading the CHAMELEON group, which investigates new hardware and software architectures for energy-efficient systems. Currently, his research interests include low-power communication, and reconfigurable architectures for energy reduction.