Cognitive Radio Design on an MPSoC Reconfigurable Platform

Qiwei Zhang, Andre B.J. Kokkeler and Gerard J.M. Smit
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands
Email: q.zhang@utwente.nl

***Abstract*— Cognitive Radio has been proposed as a promising technology to solve today's spectrum scarcity problem by dynamic spectrum access. The 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 FFT for OFDM 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.**

I. Introduction

Recent studies show that most of the assigned spectrum is underutilized. On the other hand, the increasing number of wireless multimedia applications leads to spectrum scarcity. Cognitive Radio ([1], [2]) is proposed as a promising technology to solve the imbalance between spectrum scarcity and spectrum under-utilization. In Cognitive Radio, spectrum sensing is done in order to locate the unused spectrum segments in a targeted spectrum pool. These segments will be used optimally without harmful interference to the licensed users. This technology is called *spectrum pooling* [3]. In spectrum pooling, OFDM is proposed as the baseband transmission scheme. Those subcarries which cause interference to the licensed user should be nullified. Therefore OFDM based Cognitive Radio has to be reconfigurable to be adaptive to the changing wireless channels. The reconfigurability has to be supported by a reconfigurable platform. Our research undertaken in the Adaptive Ad-hoc Freeband (AAF) project [4] focuses on mapping the baseband algorithms of Cognitive Radio onto a reconfigurable platform.

II. MPSoC 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 (GPPs) and Digital Signal Processors(DSPs) which are inadequate for future high data rate wireless communications in terms of the processing speed and the energy efficiency. With the advance of the semiconduct technology, the future trend of wireless baseband processors moves toward Multiprocessor System-on-Chips (MPSoCs) which integrate heterogeneous processing elements tailored for different processing tasks. MPSoCs offer high performance,

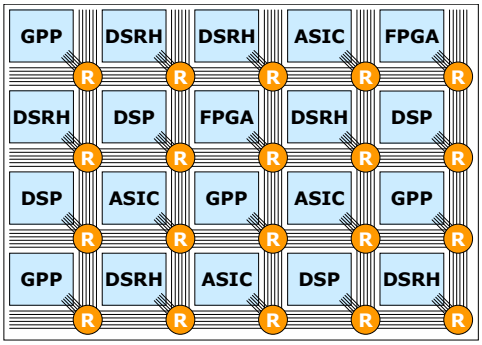

Fig. 1. Heterogeneous Multiprocessor tiled SoC

reconfigurability and energy efficiency. Therefore, we proposed a tiled MPSoC (see figure 1) architecture to support Cognitive Radio [5]. These tiles can be various processing elements including General Purpose Processors (GPPs), Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs) and Domain Specific Reconfigurable Hardware (DSRH) modules which target the specific algorithm domains. The Montium [6] tile processor developed at the University of Twente is an example of a DSRH. It targets the digital signal processing (DSP) 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.

III. TTL Design Methodology

MPSoCs offer many advantages as mentioned. 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 reuse of hardware and software modules are limited and a method for exploring their trade-offs is missing. 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 [7] 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. A task implementation is the implementation on a particular tile, e.g. object code for an ARM or configuration data for the Montium processor. 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 [8]. The TTL approach is extended for modelling reconfigurable applications such as adaptive baseband processing in Cognitive Radio.

IV. Design Case: A Low Complexity FFT for OFDM Based Cognitive Radio

In this paper, we present a design case of a reconfigurable low complexity 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.

A. 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 [5], 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:

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

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 kth 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:

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

A functional diagram of the system is presented in Figure 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 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

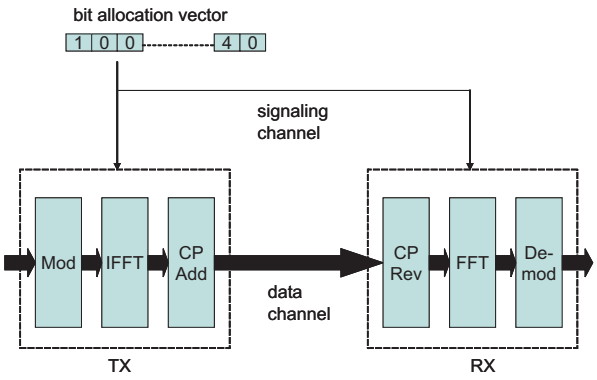

Fig. 2. OFDM for Cognitive Radio

basic idea is to load more bits on good subcarriers and load zeros to carriers which cause interference to the licensed user or lead to poor transmissions. The expectation is that

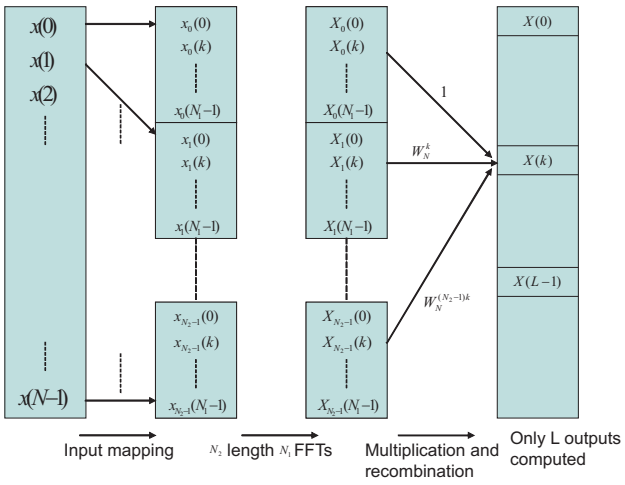

Fig. 3. Computational structure of transform decomposition

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.

B. 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 proposed a low complexity FFT algorithm as an option for OFDM based Cognitive Radio in [9]. We call this FFT algorithm *sparse FFT*. The algorithm is based on transform decomposition in [10], 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 [10], here we only show the computational structure in figure 3. We made some modifications to the original algorithm 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 output. 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 $n_2 = 1, 2, ..., N_2$. According to the computational structure, we make quantitative analysis on the computational complexity by counting the number of complex multiplications. So the number of multiplication for sparse FFT is $(N_2 - 1) * L + \frac{N}{2} \log_2 N_1$,

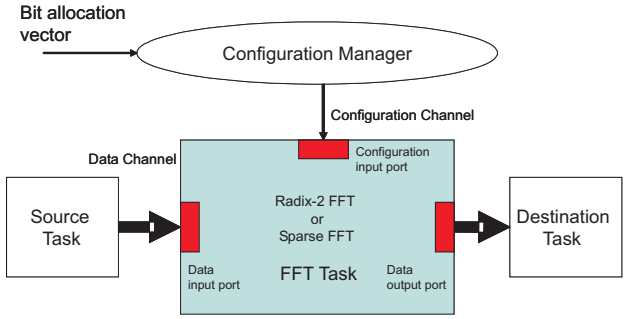

Fig. 4. Task graph of reconfigurable sparse FFT

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.

C. 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.

At the first step, we create the task graph (see figure 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 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. The FFT task either performs radix-2 FFT or the sparse FFT depending on L.

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)
 {\\acquire 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 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.

D. Results

We applied the reconfigurable sparse FFT to an OFDM receiver based on the specification in [11] for the AAF system.

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

TABLE I

OFDM PARAMETERS: SAMPLE FREQUENCY AND SYMBOL DURATION.

The OFDM parameter set under consideration is given in Table I. The bandwidth for the OFDM system is 5.12MHz and there are 512 subcarriers in one OFDM symbol. Thus, subcarrier spacing Δf is 10kHz 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 figure 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 with 1. In Scenario 1, 420 of 512 indexes are non-zeros which means most of the subcarriers can be used by Cognitive Radio. To avoid interference to an emerging 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

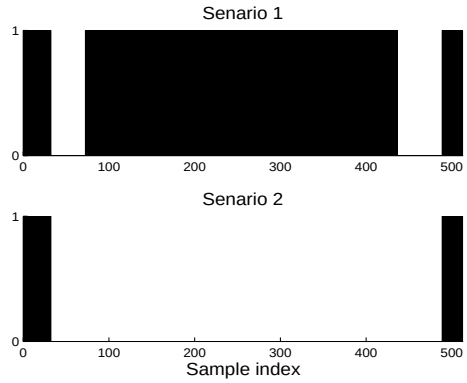

Fig. 5. Example of reconfiguration

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 II shows the computation workload of the FFT task in two scenarios generated by TTL. The reduction of

	complex multiplication (instruction count)
S_1	2304
S_2	1928

TABLE II

COMPUTATION WORKLOAD OF THE FFT TASK

computation in Scenario 2 is due to the sparse FFT which takes advantage of sparse data. Considering the worst case execution time (WCET) 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 III. The profile information generated by the TTL run-time environment is independent of platforms. However, it can help to generate the platform dependent profile

	complex multiplications (per second)
S_1	23×10^6
S_2	19×10^6

TABLE III

MINIMUM PROCESSING REQUIREMENTS

for specific implementations. By associating execution times with instructions, and by multiplying these execution times with the instruction counts, one can get 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 III, we find that the Montium has to run at at least 23MHz for Scenario 1 and 19MHz for Scenario 2. In

other words, the sparse FFT will save 16% processing capacity. The reconfiguration is only done when the bit allocation vector has been updated. We expect the bit allocation vector not to change very often, at least it is constant during 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. The result in figure 6 shows that the computation workload increases with the number of non-zero L.

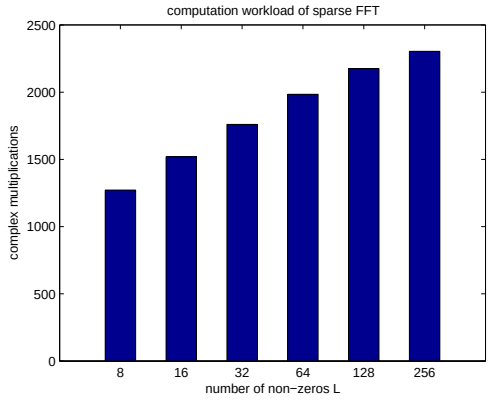

Fig. 6. Computation workload of sparse FFT for 512 samples

V. 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.

VI. Conclusion

In this paper we present a system level design methodology for mapping Cognitive Radio onto an MPSoC platform. The design methodology is based on a task transaction level interface (TTL) to partition the application into communicating tasks. By making the communication explicit, the computation (task) is implemented separately from 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 has given 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. Design trade-offs can be made in an early design stage.
- TTL is suitable for modelling a reconfigurable application.
- The TTL code is portable to the reconfigurable platform.

Therefore, 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.

Acknowledgment

We acknowledge the support for TTL 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 (IRCT) of the Technical University Delft (TUD) and the Signal and System (SAS) group at the University of Twente (UT) for the fruitful discussions in the AAF project.

References

[1] J. Mitola III. *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, PhD Thesis, Royal Institute of Technology, Sweden, May. 2000.

[2] S. Haykin *Cognitive radio: Brain-empowered wireless communication*, IEEE J. Select. Areas Commun., vol. 23, no.2:pp 201-220, Feb. 2005.

[3] T.A. Weiss and F.K. Jondral *Spectrum pooling: An innovative strategy for the enhancement of spectrum efficiency*, IEEE Commun. Mag., Mar. 2004

[4] www.freeband.nl

[5] Qiwei Zhang, Andre B.J. Kokkeler and Gerard J.M. Smit *A Reconfigurable Radio Architecture for Cognitive Radio in Emergency Networks*, European Conference on Wireless Technology, September 2006, Manchester, UK

[6] Paul Heysters *Coarse-Grained Reconfigurable Processors; Flexibility meets Efficiency*, PhD Thesis, University of Twente, Sep. 2004

[7] Pieter van der Wolf et al. *Design and Programming of Embedded Multiprocessors: An Interface-Centric Approach*, In Proceedings of ISSS+CODES, Sept. 2004

[8] Qiwei Zhang, Andre B.J. Kokkeler and Gerard J.M. Smit *Adaptive OFDM System Design For Cognitive Radio*, In: 11th International OFDM-Workshop 2006, Aug. 2006, Hamburg, Germany

[9] Qiwei Zhang, Andre B.J. Kokkeler and Gerard J.M. Smit *An Efficient FFT Implementation For OFDM Based Cognitive Radio On A Reconfigurable Architecture* acceptted by IEEE CogNet 2007 Workshop, UK

[10] Henrik V. Sorensen and Sidney Burrus *Efficient Computation of the DFT with Only a Subset of Input or Output Points*, IEEE Trans. on Signal Processing, Mar. 1993

[11] Hoeksema, F.W., Heskamp, M., Schiphorst, R. and Slump, C.H. *A Node Architecture for Disaster Relief Networking* Proceedings of the first IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN2005), November 8-11, 2005, Baltimore, USA