A System-level Design Method for Cognitive Radio on a Reconfigurable Multi-processor Architecture

Qiwei Zhang, Andre B.J. Kokkeler and Gerard J.M. Smit
Electrical Engineering, Mathematics and Computer Science Department
University of Twente
7500 AE Enschede, The Netherlands
Email: q.zhang@utwente.nl

Abstract—The future trend of software defined radio (SDR) platforms moves toward reconfigurable Multiprocessor System-on-Chips (MPSoCs). However, there is a gap between the modelling of the dynamic radio applications and the optimized implementation of the application on reconfigurable multiprocessor architectures. We aim to close this gap by applying a system level design method for the modelling and implementation of the applications on an MPSoC. The state-of-the-art radio technology based on SDR, Cognitive Radio, is considered as a design case to demonstrate the effectiveness of this method.

I. INTRODUCTION

The traditional software defined radio (SDR) 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 processing speed and energy efficiency. With the advance of the semiconductor technology, future wireless baseband processors move toward Multiprocessor System-on-Chips (MPSoCs) which integrate heterogeneous processing elements tailored for different processing tasks. MPSoCs offer high performance, reconfigurability and energy efficiency. Therefore, a tiled MPSoC (see figure 1) architecture

Fig. 1. Heterogeneous Multiprocessor tiled SoC

is proposed to support the digital processing of SDR. The tiles can be various processing elements including: General Purpose Processors (GPPs), embedded Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs) and Domain Specific Reconfigurable Hardware (DSRH) modules which target specific algorithm domains. The Montium [1] tile processor developed at the University of Twente is an example of a DSRH. It targets the digital signal processing (DSP) algorithm domain with the flexibility

to adapt to different algorithms in an energy-efficient manner. Therefore, the Montium tile processor is the key element in our proposed reconfigurable platform. 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.

However, it is a challenging task to map applications onto such MPSoCs. First, the applications to be mapped become more complex and may change their behavior dynamically. Second, applications have to be partitioned into tasks which are to be mapped on different components on an MPSoC. Therefore, designers have to deal with the low level interfaces for the inter-component communication and synchronization which may 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) [2] interface approach was proposed to help closing the design gap by raising the abstraction level. We propose to use this design method for our MPSoC based SDR platform. The paper is organized as follows. Section 2 introduces the TTL design method. The state-of-the-art radio technology based on SDR, Cognitive Radio, is introduced as a design case in section 3. The cognitive radio applications are modelled and implemented in the TTL framework. Results are presented in Section 4 to demonstrate the effectiveness of the method. The paper ends with future work and conclusions.

II. TTL DESIGN METHOD

DSP applications are often modelled as streaming task graphs which consist of logical entities like tasks, ports and channels [2]. *Tasks* are entities that perform data processing and they can communicate with other tasks by sending data via *ports* to *channels* which interconnect tasks. The TTL communication interface is based on these logical models. Inter-task communications are invoked by calling TTL interface functions, like *read* or *write* on their ports. Computation is separated from communication which makes that computation

can be implemented easier onto on particular tiles. It also provides high level profile information in terms of computation workload and communication workload. In such a way, we can explore and validate design alternatives at a high level of abstraction. Subsequently, the high level TTL specification will be mapped onto a multiprocessor architecture.

However, embedded systems are becoming more and more complex, supporting multiple use cases. Therefore, TTL has been extended to model dynamic applications [3]. Dynamic applications result in changes in the task graph. Three types of task graph changes are considered: 1) *Topology*: removing or adding tasks and channels, 2) *Binding*: tasks may be processed on different processors, 3) *Parameters*: certain parameters for tasks (e.g. radio transmission modes). The reconfiguration is realized by introducing an extra entity, a *Configuration Manager* (CM), to the TTL logical model. The CM is responsible for initiating the task graph and configuring the tasks at run-time. This feature makes TTL a suitable design method to model dynamic applications and as an interface for the MPSoC platform implementation.

III. DESIGN CASE: COGNITIVE RADIO ON RECONFIGURABLE MPSOC

We propose to use the TTL approach as a system level design method to map dynamic radio applications to our proposed MPSoC SDR platform. A highly dynamic radio application, Cognitive Radio, is considered as a design case.

A. Introduction to the Cognitive Radio application

Recent studies show that most of the assigned radio spectrum is underutilized. On the other hand, the increasing number of wireless multimedia applications leads to a spectrum scarcity. Cognitive Radio ([4], [5]) is proposed as a promising technology to solve the imbalance between spectrum scarcity and spectrum under-utilization by dynamic spectrum access. In Cognitive Radio, spectrum sensing is constantly done in order to locate the unused spectrum segments in a targeted spectrum pool [6]. In spectrum pooling, orthogonal frequency division multiplexing (OFDM) is proposed as the baseband transmission scheme. The cognition is realized by nullifying those subcarriers which cause interference to the licensed user (a user who has the legal license for the spectrum). The remaining frequency segments will be used optimally by Cognitive Radio. Cognitive Radio has to operate in different bands under various data rate and combat adversary channel conditions. Therefore, physical layer reconfigurability has to be supported by an SDR platform.

The background information of OFDM can be found in the textbook such as [7]. A generic task graph for the processing of OFDM data symbols on the receiver side is shown in figure 2. A specific OFDM system is characterized by a set of parameters. By applying different parameter settings in each task, the OFDM system can adapt to various channel conditions and provides various data rates. Therefore, the OFDM system for Cognitive Radio is a parameterizable OFDM processing chain configured by the configuration manager, see figure 3. The

Fig. 2. The task graph of an OFDM receiver

Fig. 3. OFDM for Cognitive Radio

parameters considered in our system are shown in Table I. The number of OFDM symbols per frame is limited by the channel coherence time, during which the channel characteristics are constant. The number of guard samples is chosen to deal with different channel delay spreads. Generally not all data is used to carry useful information. A part of data (e.g. pilots) is used to guarantee reliable transmissions. Different pilots are used for different purposes such as channel estimation or phase offset estimation. They can be placed in the preamble section prior to each frame or embedded in the OFDM symbol. All the information concerning the pilots is contained in a table. Modulation modes indicate the modulation type for the OFDM samples which carry useful information. The modulation mode can be the same in one OFDM symbol but it can also differ on subcarrier basis. In the Cognitive Radio case, the modulation mode can be set to zero to nullify carriers. A format table contains the information on the organization of the data frame, the preamble and the pilots.

B	bandwidth of OFDM system
N_{sym}	number of OFDM symbols per frame
$N_{preamble}$	preamble length per frame
N	number of OFDM samples per symbol
N_g	number of guard samples per symbol
N_{data}	number of useful data per symbol
mod	modulation modes
tab_{pilot}	table for pilot information
tab_{format}	table for format information

TABLE I

THE PARAMETER SET FOR THE PARAMETERIZABLE OFDM

B. TTL implementations

Based on the OFDM parameters of our Cognitive Radio system in [8], we modelled the OFDM processing chain on the receiver side (see figure 2) in the TTL framework. The tasks are implemented in C/C++ and inter-task communications are function calls from the TTL library. The CM is added as a process on top of the task graph. The parameters for the OFDM tasks are set by the CM and sent via the configuration

channel to the tasks. Each task can read the parameters from the configuration port. To demonstrate the approach, we give a pseudo code example: the frequency offset correction task for processing one OFDM data frame.

```
Task Freq_cor
{initialization;
while(true)
{local variables;
  \*processing preparation*\
  \check the configuration updates
  tryAcquiredata_int(Task_Freq_cor->config_inport){
  \read in the number of symbols in one frame
  ttl_read(Freq_cor->config_inport, num_symbol);
  \read in the number of samples in one symbol
  ttl_read(Freq_cor->config_inport, num_sample);
  }
  \read in the inverse offset factor
  for (i=0; i<num_sample; i++)
  ttl_read(Freq_cor->config_inport, cor_fac[i]);
  \*processing task*\
  for (j=0; j<num_symbol; j++)
  {\read in samples
   for (i=0; i<num_sample; i++)
   ttl_read(Freq_cor->data_inport, i_buffer[i]);
   \frequency offset correction
   for (i=0; i<num_sample; i++)
   o_buffer[i]=i_buffer[i]*cor_fac[i];
   \write out samples
   for (i=0; i<num_sample; i++)
   ttl_write(Freq_cor->data_outport, o_buffer[i]);
  }
}
}
```

The frequency offset correction task corrects the frequency offset introduced in the channel by multiplying all input samples with an inverse offset. First the basic system parameters from the configuration manager are read via the configuration port: the number of OFDM symbols in one data frame *num_symbol* and the number of samples in each OFDM symbol *num_sample*. The inverse offset, determined with the aid of pilots in the preamble prior to the data frame, is sent by the configuration manager to the frequency offset correction task. Then the OFDM samples are read and processed: complex multiplications of the data samples with the inverse offset. After processing, the samples are sent to the FFT task via the data channel.

IV. RESULTS

The TTL implementation of the whole OFDM receiver in C/C++ can run on a Linux PC and we can verify the functional correctness of parameterizable OFDM at system level. 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. The major system parameters [8] are shown in Table II. The system can operate in two different sampling rates f_s (bandwidth B) and the number of samples N in one OFDM symbol is 128 or 512. The subcarrier spacing is 10kHz and the useful symbol duration T_u is 100 μ s for both

$B = f_s$ [MHz]	N	Δf [kHz]	T_u [μ s]
5.12	512	10	100
1.28	128	10	100

TABLE II
OFDM PARAMETERS: SAMPLE FREQUENCY AND SYMBOL DURATION.

bandwidths. To cope with different delay spreads, the ratio between the number of guard samples N_g and the number of OFDM samples N can be set to 1:4, 1:8, 1:16, 1:32. As a working assumption, one OFDM data frame contains 25 OFDM data symbols thus one frame duration is less than the coherence time (4-5 ms in the considered frequency 400-800MHz) of the channel. The frequency offset and the channel equalization coefficients are determined in the preamble prior to each OFDM frame and updated on a frame basis. The pilots in the OFDM symbols are used for the phase offset correction. We define the ratio of the useful data carriers to the number of total carriers $N_{data}:N$ as 3:4. Modulation modes are chosen from BPSK, QPSK, 16QAM, 64QAM and zero. The zero modes are used to nullify the subcarrier. If a moderate code rate 3/4 is applied for the channel coding, the maximum achievable data rates (when the 64QAM mode is used) for $B=5.12$ MHz and $B=1.28$ MHz are 16Mb/s and 4Mb/s respectively. Suppose that Cognitive Radio use only 25% of the data carriers in order to avoid the licensed user. The system still may achieve 4Mb/s and 1Mb/s under $B=5.12$ MHz and $B=1.28$ MHz.

We made a computation workload analysis based on the TTL model. The instructions for the complex multiplication are annotated for analysis because they are the major contributors to the computational complexity. The computation workload for processing one OFDM data frame is shown in Table III based on the given system parameters. The demap task is not included in the table because no complex multiplications are involved and only decision and look-up table operations are performed. The computation workload increases significantly if the system parameter N (the number of OFDM samples) changes from 128 to 512. The change of guard samples N_g results in a workload change for the guard time removal task due to the changing length of the correlation window. From the table we can see the FFT task is the most computationally intensive task. Considering that the worst case execution time (WCET) of the system should be less than the symbol duration 100 μ s, the system has to be able to compute a 512-point FFT which needs 2304 complex multiplications within 100 μ s. Therefore the minimum processing capacity required by the parameterizable OFDM system is 23×10^6 complex multiplications per second.

The profile information provided by the TTL run-time environment is platform independent. 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 get a rough estimate of the total execution time

$N : N_g$	T_g rem.	Freq. cor.	FFT	Channel eq.	Phase offset cor.
128 : 32	800	3200	11200	3200	2425
128 : 16	400	3200	11200	3200	2425
128 : 8	200	3200	11200	3200	2425
128 : 4	100	3200	11200	3200	2425
512 : 128	3200	12800	57600	12800	9625
512 : 64	1600	12800	57600	12800	9625
512 : 32	800	12800	57600	12800	9625
512 : 16	400	12800	57600	12800	9625

TABLE III

COMPUTATION WORKLOAD FOR EACH OFDM FRAME (25 OFDM SYMBOLS) IN TERMS OF THE NUMBER OF COMPLEX MULTIPLICATIONS

f_{clk} 100MHz	T_g rem.	Freq. cor.	FFT	Channel eq.	Phase offset cor.
Exec. (μs)	32	128	576	128	96

TABLE IV

EXECUTION TIME ON THE MONTIUM FOR ONE OFDM FRAME
($25 \times 125 \mu s = 3.125 ms$)

of a task on a certain processor. A Montium processor can execute one complex multiplication instruction in one clock cycle. Therefore we can easily derive the profile information for processing one OFDM frame on the Montium. Taken the parameter set $N = 512$ and $N_g = 128$ as an example, the profile information for one OFDM frame on the Montium running at 100MHz is derived from the TTL profile in Table IV. The power consumption of the Montium in $0.13 \mu m$ technology is estimated at 0.577 mW/MHz [1]. We can further estimate the energy consumption of each task for processing one OFDM frame, see figure 4. The most energy hungry part is the FFT task which costs more than $33 \mu J$ for processing one OFDM frame. Therefore to have an efficient FFT implementation is crucial to reduce the energy consumption of the whole OFDM baseband.

Fig. 4. The energy consumption on the Montium for one OFDM frame

V. FUTURE WORK

The algorithms for the Cognitive Radio baseband can be developed, experimented and verified in the TTL framework.

The computation components can be implemented on different processors and further optimized. 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 method for mapping the Cognitive Radio application onto an MPSoC platform. The design methodology is based on a task transaction level (TTL) interface to partition the application into communicating tasks. By making the communication explicit, the computation (task) is implemented separately from communication. The parameterizable OFDM for Cognitive Radio is used as a design case to show the effectiveness of the TTL method. From the design case, the TTL method gives the following advantages:

- The TTL model helps to generate a task based design rather than a sequential code which is not suitable for multi-processor architectures. A task can have processor specific implementations and these implementations are separated from inter-task communications. The design, based on the TTL, can be verified at system level.
- The profile information generated by the TTL run-time environment can be used for computation and communication complexity analysis at a system level. The design trade-offs can be made in an early design stage.
- The TTL is suitable for modelling a reconfigurable application.

In conclusion, the TTL is a suitable system level design method to map the dynamic radio application, namely Cognitive Radio, onto an MPSoC SDR platform.

ACKNOWLEDGEMENTS

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 Technical University Delft (TUD) and the University of Twente (UT) for the fruitful discussions in the AAF project.

REFERENCES

- [1] Paul Heysters, Coarse-Grained Reconfigurable Processors; Flexibility meets Efficiency, PhD Thesis, University of Twente, Sep. 2004
- [2] Pieter van der Wolf et al. Design and Programming of Embedded Multiprocessors: An Interface-Centric Approach, In *Proceedings of ISSS+CODES*, Sept. 2004
- [3] Jeffrey Kang et al. An Interface For the Design and Implementation of Dynamic Applications on Multi-processor architectures, In *Proceedings of ESTmedia*, 2005
- [4] J. Mitola III., Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio, PhD Thesis, Royal Institute of Technology, Sweden, May. 2000.
- [5] S. Haykin, Cognitive radio: Brain-empowered wireless communication, *IEEE J. Select. Areas Commun.*, vol. 23, no.2:pp 201-220, Feb. 2005.
- [6] T.A. Weiss and F.K. Jondral, Spectrum pooling: An innovative strategy for the enhancement of spectrum efficiency, *IEEE Commun. Mag.*, Mar. 2004
- [7] Richard Van Nee and Ramjee Prasad, OFDM for Wireless Multimedia Communications, Artech House Publisher
- [8] 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