Reconfiguration-aware Spectrum Sharing for FPGA based Software Defined Radio

Hessam Kooti, Elaheh Bozorgzadeh, Shenghui Liao, Lichun Bao
Computer Science Department
University of California, Irvine, CA, USA
{hkooti,eli,shenghul,lbao}@ics.uci.edu

Abstract —**This paper focuses on reconfigurable systems for software defined radio applications in which the underlying hardware is dynamically reconfigured for packet processing of multiple protocols. However, due to non-negligible reconfiguration delay overhead, the physical layer may not be able to respond to all the packets scheduled by MAC layer. In this paper, we present a reconfiguration-aware spectrum sharing and spectrum access scheduling at MAC layer. We considered four protocols similar to WiFi, WiMax, GPRS, and WCDMA on a FPGA-based system. Our results show that the optimal solution outperforms the adopted existing heuristic for time slot scheduling by 13.29%.**

Keywords- Spectrum Sharing, Hardware Reconfiguration, Real-time Scheduling

I. INTRODUCTION

In software defined radios (SDRs) [1], a wide variety of strategies and protocols can be implemented in software and the same spectrum can be shared. Hardware reconfiguration is a promising platform to implement SDR [2,3], which enables implementation of different protocols on a chip and feature changes in the hardware due to communication protocol update. In this paper, we focus on multiple protocol support on FPGA-based reconfigurable systems.

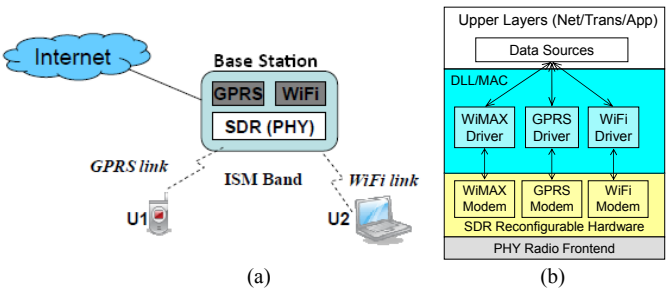
The main drawback of reconfigurable hardware platform is the reconfiguration overhead between different protocols. We propose to plan ahead at MAC layer for reconfiguration overhead at physical layer, and hence, to avoid unexpected throughput degradation caused by such overhead. We focus on spectrum access scheduling at the MAC layer, where time slots are allocated to different protocols for packet processing at the physical layer. If we do not consider this overhead during spectrum sharing, the hardware may not be ready when the MAC layer needs to send/receive a packet for a different protocol. Hence, some packets may be dropped and this may lead to significant system throughput degradation. In [4], an FPGA based platform is proposed for implementing an SDR device capable of communication over 802.11 and WCDMA networks. In [5], the authors implement a system which can change functionality from GSM to EDGE by runtime partial reconfiguration of FPGA devices. In [6], the authors exploit reusing modules to avoid reconfiguration overhead. In [7], the authors target reducing reconfiguration for a set of scheduled tasks by floorplanning all of them simultaneously. To the best of our knowledge, this paper is the first paper considering this overhead during spectrum scheduling in SDR application.

The main contribution in this paper is scheduling the time slots for packet processing of different protocols on the reconfigurable hardware, while considering the reconfiguration overhead between implementation of different protocols. We present a Mixed Integer Linear Programming (MILP) formulation for this problem. The proposed scheduling maximizes the number of feasible time slots MAC layer allocates to protocols (best effort). If all the time slots were schedulable, the scheduler will minimize the reconfiguration overhead between protocol implementations. We considered four protocols similar to WiFi, WiMax, GPRS, and WCDMA on a FPGA-based system. In comparison with modified EDF algorithm, our results show that the optimal solution outperforms the adopted existing heuristic for time slot scheduling by 13.29%.

II. SDR APPLICATION

Figure 1-a illustrates the hardware and software elements of a base station using the SDR. *Spectrum access scheduling* is the approach that is used in SDR to share the spectrum between different protocols. In this approach, the system is aware of inter-operation and coexistence of different wireless users from the beginning.

Figure 1-b illustrates the overall system architecture that supports coexistence of heterogeneous wireless communication systems, e.g. WiFi, GPRS, and WiMax. Various non-time stringent data link layer protocols run in the software portion of the SDR platform, while the hardware portion implements the time stringent and computationally intensive modulation/demodulation (modem) functions. Since the topper layer is not time



(a) (b)

Figure 1: a) Software Defined Radio elements b) The Base-Station HW/SW Architecture for Spectrum Access Scheduling Based on SDR

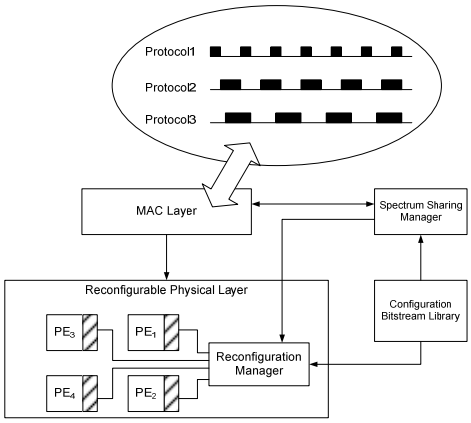


Figure 2: Modified SDR Application

stringent, it is the job of our spectrum access scheduler to schedule the protocols and make it time stringent for the physical layer. In addition, the radio front-end installs frequency dependent antenna segments.

Figure 2 shows the physical layer implementation for SDR. MAC layer sends stream of data packets from different protocols to physical layer. MAC layer assigns some time slots to protocols so they can share the same spectrum. In a regular SDR, when MAC layer assigns time slots, it is not aware of lower hardware (physical layer) and the reconfiguration overheads in that layer. Hence many time slots assigned to protocols may be missed because the hardware is not ready for execution. When time slots are missed, the packets cannot be sent and they should be sent later by MAC layer. If this procedure is repeated, it reduces the performance of the system. We need a method to give some feedbacks from SDR reconfigurable hardware part to MAC layer. So we add the *Spectrum Sharing Manager* to SDR. This block receives the configuration bitstream of different protocols from *Configuration Bitstream Library*. Based on this information spectrum sharing manager calculates the time needed to switch from each protocol to another one. Using these overheads and the timings from the MAC layer, spectrum sharing manager will make decision about three problems: 1-The order of execution of protocols. 2- The implementation that should be used for each protocol (we assume that each protocol has several implementations) 3- Delaying the packet processing if needed to increase the number of feasible communications.

In this paper we target a heterogeneous platform consisting of several Processing Elements (PEs). These PEs can be implemented on single or multiple FPGAs (Figure 2). For each PE, there is a reconfigurable part which is reconfigured/updated during the runtime. Based on the timings from spectrum sharing manager, the *Reconfiguration Manager* forces the PEs to reconfigure appropriately.

The SDR application in this paper is fully streaming and there is no data dependency between the protocols. We model the transition overhead between protocols in three categories: ***1) Fixed overhead:*** This type of overhead is a fixed overhead per protocol on all PEs. ***2) Protocols initiation overhead:*** This type of overhead depends both on the protocol and the PE. ***3) Inter-protocol overhead:*** This type of overhead depends on PE and the current protocol as well as the next incoming protocol. Incorporating transition overhead during spectrum access scheduling is our contributions in this work.

During spectrum sharing in MAC layer we can delay transmission time of some protocols in order to reduce the time overlap among different protocols. However, this action should not lead to delaying the deadline of the protocol instance beyond its period. In this improvement, we do not change the start time of the periods, but only delaying the execution of a protocol instance within its interval. The same delay applies to all the instances of a protocol.

III. Spectrum Sharing under Transition Overhead

We have a set of PEs (K) and we are given a set of periodic protocols, $\tau = \{\tau_i\}$. During spectrum sharing, we should schedule the time slots allocated to each protocol on the PEs. We can assume each protocol is represented by a task. Hence, we formulate the spectrum sharing problem as a real-time task scheduling problem in which duration of each time slot is equal to execution time of the corresponding task. Each periodic protocol τ_i is characterized by a triplet (p_i, d_i, E_i). p_i is the period, d_i is the relative deadline, and E_i is a set of execution times of protocol τ_i (duration of time slots for each protocol) on each PE, $E_i = \{e^r_{i,m} \mid r \in K, m \in i-conf\}$. *i-conf* is the set of configurations for protocol τ_i. If all the PEs are identical, we refer to execution time as $e_{i,m}$. The inter-protocol overhead between the *m-th* configuration of protocols τ_i and *n-th* configuration of τ_j on PE *r* is $s^r_{i,m;j,n}$ and protocol initiation overhead of *m-th* configuration protocol τ_i on PE *r* is $o^r_{i,m}$.

We present each instance of the periodic protocols with a node in a directed graph and each node is associated with corresponding time window of the protocol. Figure 3 shows a graph representation for 3 nodes, each of them with 3 configurations. Each directed edge indicates a possible reconfiguration between two consecutive implementations. There is a start time t_i associated with each node *i* in the graph. As long as the start time of each node lies within the corresponding time window (i.e. $[a_i, b_i]$), the schedule of that node is feasible. Hence, the following conditions need to be met for two consecutive nodes in Figure 3 if the highlighted edge is chosen.

$$a_{A,1} \le t_{A,1} \le b_{A,1} \;\&\; a_{B,3} \le t_{B,3} \le b_{B,3} \;\&\; t_{A,1} + e_{A,1} + s_{A,1;B,3} + o_{B,3} \le t_{B,3}$$

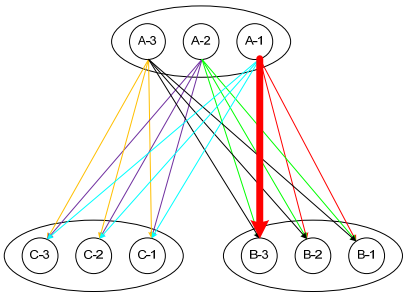


Figure 3: Graph representation of multi-configuration protocols

We formulate the problem over the period of the whole system referred to as P_{Total} , by computing the least common multiple of periods of input protocols. We present a MILP for the proposed spectrum sharing problem. Within P_{Total}, each protocol instance is represented by a node. For *j-th* instance of τ_i the time frame of corresponding node is $[(j-1)\times p_i, (j-1)\times p_i + d_i - e_i]$.

We have a set of nodes representing the protocol instances as $N=\{1,2,...,n\}$. Set $N^+=\{0,1,2,...,n+1\}$ includes set N and the initial and end nodes (dummy nodes for initial state which are the same node). The set $K=\{1,2,...,m\}$ represents the set of PEs. Set A^+ is the set of all the transitions in our system (i.e. $A^+ = N^+ \times N^+$). The inputs and variables are shown in Table I and Table II.

TABLE I: INPUTS OF ALGORITHM

Input	Description
$s^r_{i,m\ j,n}$	transition overhead from *m-th* configuration of τ_i to *n-th* configuration of τ_j on PE *r*
$o^r_{i,m}$	protocol initiation overhead of *m-th* configuration of protocol τ_i on PE *r*
p_i	period of τ_i
d_i	relative deadline of τ_i
$e^r_{i,m}$	execution time of *m-th* configuration of τ_i on PE *r*
K	set containing all the PEs

TABLE II: VARIABLES OF ALGORITHM

Variable	Description
st_i	start time of τ_i
$x^r_{i,m\ ;\ j,n}$	binary variable which indicates that if there is any transition between *m-th* configuration of node *i* and *n-th* configuration of node *j* on PE *r* or not
$y^r_{i,m}$	binary variable which indicates that if *m-th* configuration of node *i* is implemented on PE *r* or not
z^r_i	binary variable which shows that if node *i* is implemented on PE *r* or not
$w^r_{i,m}$	variable indicating the improved time slot duration of *m-th* configuration of node *i* implemented on PE *r*
$t^r_{i,m}$	start time of *m-th* configuration of node *i* on PE *r*

Equations (1), (2) and (3) ensure that *at most* one configuration of each node is executed on PEs. Equation (4) guarantees that after implementing each node, we will implement another node and the sequence will continue. Equation (5) indicates that all the PEs are considered in our solution. For delayed transmission, we compute the valid range for start time of each node with Equations (6), (7) and (8). Equation (9) guarantees that when two consecutive nodes are implemented on the same PE, the second one would not start until execution of the first one is over. Equation (10) ensures that the start time of each node is within the valid interval. *M* is an arbitrary large constant. We outline two objectives that are related to the proposed application in this paper. First objective is to maximize the total number of nodes scheduled on PEs (best effort); hence, it is a feasibility objective (11). Second objective in Equation (12) minimizes the transition overhead if all the nodes are scheduled on PEs. More details about these equations are presented in [8].

(1) $\sum_{j\in N^+ n\in j-conf} x^r_{i,m\ ;\ j,n} = y^r_{i,m}\ , i\in N, m\in i-conf, r\in K$

(2) $\sum_{m\in i-conf} y^r_{i,m} = z^r_i\ ,\ i\in N\ ,\ r\in K$

(3) $\sum_{r\in K} z^r_i \le 1\ ,\ i\in N$

(4) $\sum_{i\in N^+}\sum_{m\in i-conf} x^r_{i,m\ ;\ h,p} - \sum_{j\in N^+}\sum_{n\in j-conf} x^r_{h,p\ ;\ j,n} = 0\ , h\in N, p\in h-conf, r\in K$

(5) $\sum_{i\in N^+}\sum_{m\in i-conf} x^r_{0\ ;\ i,m} = 1\ \ and\ \ \sum_{i\in N^+}\sum_{m\in i-conf} x^r_{i,m\ ;\ (n+1)} = 1\ \ , r\in K$

(6) $0 \le st_i \le p_i - d_i\ , i\in\tau$

(7) $a_{u,m} = (j-1)\times p_i + st_i\ , i\in\tau, u\in N, m\in u-conf$

(8) $b^r_{u,m} = (j-1)\times p_i + d_i - w^r_{i,m} + st_i\ \ , i\in\tau, u\in N, m\in u-conf$

(9) $t^r_{i,m} + o^r_{i,m} + w^r_{i,m} + s^r_{i,m;j,n} - M(1-x^r_{i,m;\ j,n}) \le t^r_{j,n}$
$, (i,j)\in A^+, m\in i-conf, n\in j-conf, r\in K$

(10) $a_{i,m}\, y^r_{i,m} \le t^r_{i,m} \le b^r_{i,m}\, y^r_{i,m}\ , i\in N, m\in i-conf\ , r\in K$

(11) $\max \sum_{i\in N}\sum_{r\in K} z^r_i$

(12) $\min \sum_{r\in K}\sum_{(i,j)\in A}\sum_{m\in i-conf}\sum_{n\in j-conf} s^r_{i,m\ j,n} \times x^r_{i,m\ ;\ j,n}$

Since we formulate the problem as a real-time task scheduling problem, we also considered integrating the reconfiguration overhead as a transition overhead in the existing solutions for real-time task scheduling. We modified commonly used EDF (Earliest-Deadline-First) algorithm under transition overhead. Also, we considered an optimal task scheduling under transition overhead as a fixed value. In the latter case, we considered the worst reconfiguration overhead per protocol as the fixed overhead.

IV. EXPERIMENTS

A) Experiment Setup: We do our experiments on the physical layer implementation for 4 different protocols. These protocols are very similar to WiFi (Protocol A), WiMax (Protocol B), GPRS (Protocol C) and WCDMA (Protocol D). A set of distinctive and important components of these protocols like Turbo Enc/Dec, RS Enc/Dec, FFT/IFFT, etc are implemented on the target FPGA.

The FPGA architecture that we have assumed for our benchmarks is Virtex 4 architecture. Reconfiguration time overhead includes the time ICAP fetches the configuration bits and reconfigures the FPGA device. This delay for one configuration frame is the same across all V4 devices and is equal to 13.1us for one frame.

In this paper we assume multiple implementations for protocols. We design two floorplans for each protocol. We apply hardware reuse and configuration prefetch during the reconfiguration overhead computations. We have 8 scenarios in this case study. 6 scenarios have 3 protocols and 2 FPGAs and 2 of them have 4 protocols and 3 FPGAs. Table III shows the timings we used for protocols in our simulations.

TABLE III: PROTOCOLS USED IN OUR SIMULATIONS

Protocols	Allocated time slots (msec)	Period (msec)	Deadline (msec)
WiFi (FDMA)	A number from set {3.1, 4,5.1 }	A number from set {5.9, 6.5, 10}	A number from set {5, 5.5, 8 }
WiMax (FDMA)	Same duration for Uplink/Downlink a number from set {1.2, 1.5, 2, 3}	4.615 (An Uplink or Downlink subframe During each period)	4.6 (Deadline less than period, since small times are needed for RTG and TTG)
GPRS (TDMA)	Two GPRS time slots (1.032)	4.615	1.152
WCDMA (CDMA)	A number from set {3, 4}	10	A number from set {7, 8}

We implemented our MILPs using CPLEX solver. We modified non-preemptive EDF algorithm [9] to consider transition overhead and also implemented optimal spectrum sharing with fixed maximum transition overhead (i.e. $f_{j,n} = \max_{i \in N, m \in i-conf, r \in K} \left(S^r_{i,m;j,n} \right)$ is the fixed overhead before *n-th* configuration of node *j*). We simply include this overhead in the time we allocated to protocols during scheduling.

We ran two sets of experiments on these protocols. In the first set, we used our method for best effort to maximize the number of packets for processing (i.e. maximum feasibility). In the second set, we applied our method on scenarios with 100% feasibility to minimize the total transition overhead.

B) Results: In Table IV, the percentage of allocated time slots that are feasible with each scheduling is reported. For scheduling without multi configuration, we randomly choose one of the implementations of each protocol in our simulations. By using only one of the configurations, the feasibility decreases but it is still in an acceptable range. But the results from modified EDF and maximum overhead scheduling show that using these methods is not efficient and ignoring the real values of the overheads decreases the feasibility drastically. The results show on average 13.29% and 14.31% improvement in comparison with modified EDF and maximum overhead scheduling, respectively.

TABLE IV: BEST EFFORT OBJECTIVE

	Our Algorithms			Existing Algorithms	
Scenario	Multi Conf. with Param. Input	Single Conf. with Param. Inputs	Single Conf. without Param. Inputs	modified EDF	Maximum Overhead
a	91.67%	90.28%	87.5%	66.67%	65.28%
b	100%	100%	100%	68.06%	79.17%
c	100%	100%	100%	98.41%	100%
d	100%	100%	100%	100%	100%
e	100%	100%	93.06%	84.72%	72.22%
f	100%	100%	100%	100%	100%
g	96.51%	93.02%	90.7%	76.74%	81.4%
h	100%	100%	96.51%	87.21%	75.58%
Average	98.52%	97.91%	95.97%	85.23%	84.21%

The results for minimum transition overhead objective are shown in Table V. This simulation is only done for the scenarios which implementing all the allocated time slots were feasible and gray boxed are the cases in which all the time slots were not feasible. There is a significant difference between our scheduling and modified EDF. Transition overhead in modified EDF is almost 6 times greater than our scheduler. Also the total transition overhead in scheduling with maximum overhead is in the order of modified EDF. So ignoring the actual values of the overheads and using a fixed value instead increases the total transition overhead by 5 times on average.

TABLE V: MINIMIZING OVERHEAD OBJECTIVE

	Our Algorithms			Existing Algorithms	
Scenario	Multi Conf. w/ Param. Inputs	Single Conf. w/ Param. Inputs	Single Conf. w/o Param. Inputs	modified EDF	Maximum Overhead
a					
b	29.3	34.65	38.4		
c	6.38	6.38	6.38		33.82
d	5.8	5.8	5.8	32.53	26.81
e	13.35	15.94			
f	5.85	5.85	5.85	34.35	28.06
g					
h	13.61	15.09			

V. CONCLUSION

This paper focuses on spectrum sharing of SDR application systems on reconfigurable hardware platform, where runtime reconfiguration incurs non-negligible delay overhead during configuration. We propose a MILP formulation for this problem. The results show 13.29% improvement in comparison with modified EDF and 14.31% in comparison with maximum overhead scheduling in terms of feasibility (best effort). Also our algorithm can make the total transition overhead 6 times and 5 times smaller than total transition overhead when using modified EDF algorithm and maximum overhead scheduling, respectively.

REFERENCES

[1] J. Mitola, "Software Radio Architecture: A Mathematical Perspective," IEEE Journal on Selected Areas in Communications, 1999, pp. 514-538.

[2] Y. Lin, H. Lee, M. Woh, Y. Harel and S. Mahlke, "SODA: A High-Performance DSP Architecture for Software-Defined Radio," IEEE Micro, 2007.

[3] P. Murphy, A. Sabharwal and B. Aazhang, "Design of warp: A wireless open-access research platform," European Signal Processing Conference Program, 2006.

[4] H. Harada, "Software defined radio prototype for W-CDMA and IEEE802.11a wireless LAN," IEEE Vehicular Technology Conference, 2004.

[5] J.P. Delahaye, G. Gogniat, C. Roland and P. Bomel, "Software radio and dynamic reconfiguration on a DSP/FPGA platform," Frequenz, Journal of Telecommunications, vol. 58, pp. 152-159, 2004.

[6] S. Ghiasi and M. Sarrafzadeh, "Optimal Reconfiguration Sequence Management," Asia and South Pacific Design Automation Conference, 2003.

[7] L. Singhal and E. Bozorgzadeh, "Multi-layer Floorplanning on a Sequence of Reconfigurable Designs," Field Programmable Logic and Applications, 2006.

[8] H. Kooti, E. Bozorgzadeh, S. Liao and L. Bao, "Transition-aware Real-Time Task Scheduling for Reconfigurable Embedded Systems", Design, Automation, and Test in Europe, 2010.

[9] K. Jeffay, D. Stanat, and C. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," Real-Time Systems Symposium, 1991, pp. 129-139.