COMPUTING RESOURCE MANAGEMENT FOR SDR PLATFORMS

Vuk Marojevic, Xavier Revés, Antoni Gelonch

Polytechnic University of Catalonia, C/Jordi Girona 1-3, 08034 Barcelona, Spain
{marojevic, xavier.reves, antoni}@tsc.upc.edu

Abstract – Software Defined Radio (SDR) is an emerging technology that is based on the software implementation of the signal processing blocks found in a radio transceiver. The switch between radio access technologies may then be as easy as changing the software running on a future SDR terminal. SDR terminals refer to mobile equipment and base stations. These terminals will comprise general purpose processors, digital signal processors and/or reconfigurable logic devices. As a result, typical heterogeneous computing problems may appear in the SDR context. This article focuses on the mapping issue, discusses its relevance in software defined radio, and introduces an adequate mapping algorithm. The algorithm efficiently tackles the problem of mapping SDR function chains, i.e. signal processing blocks of a SDR transceiver, to heterogeneous processing platforms. We expose our approach, discuss its performance, present extensive simulation results and derive conclusions.

Keywords – computing resource management, mapping, software define radio.

I. INTRODUCTION

The concept behind Software Defined Radio (SDR) or software radio is to implement some or all signal processing blocks of a radio transceiver in software rather than in dedicated hardware [1] [2]. As a result, programmable devices such as different types of Instruction Set Processors (ISPs) and reconfigurable logic devices, e.g. Field Programmable Gate Arrays (FPGAs) [3], may be considered for signal processing in a SDR terminal [4] [5]. By SDR terminal we, in general, refer to a Mobile Terminal (MT) as well as to a Base Station (BS) adhering to the SDR concept. One of the most relevant parts of a SDR terminal is the SDR transceiver. It implements the signal processing chain that defines the physical layer processing of a Radio Access Technology (RAT). Such a processing chain can be decomposed in functions. Each function comprises one or more processes in order to provide a proper functionality description of a signal processing block, such as (de)modulation, (de)coding, or equalization.

SDR transceivers differ from one radio access technology to another, and may also differ within the same RAT. The latter is due to manufacturer's preferences, Quality of Service (QoS) parameters and channel conditions, amongst others. Moreover, the function chain of a single-user transceiver (MT) differs from the function chain of its multi-user counterpart (BS). Finally, the practical transceiver may comprise two function chains, one for the transmitter and one for the receiver, possibly with some common building blocks.

The following example clarifies the concepts we have just introduced. A mobile subscriber with a SDR-MT from manufacturer *A* has an agreement with service provider *B* that allows him to dynamically switch between radio access technologies. The subscriber may then switch from one RAT to another as a function of the available RAT-QoS. Another reason for a RAT switch would be that the one in use is not supported in a region the subscriber sojourns. Let us suppose that the principal radio standard is UMTS (Universal Mobile Telecommunication System). When entering a building or a subway station it could be possible and necessary to switch to WLAN (Wireless Local Access Network), whereas in geographical regions without UMTS coverage, a switch to GSM (Global System for Mobile communication) could probably be fulfilled. As a consequence, it may be necessary to dynamically assign the system resources that are available on the MT to a number of different function chains. Over time, new and quite different RATs may appear together with new function chains. Additionally, processing blocks may be modified or replace by new ones.

System resources that are required by these function chains are basically computing and communication resources. Resources of a radio terminal are and will always be limited, mainly due to battery power and cost. Hence, they have to be assigned wisely to competing functions or processes. Processing requirements are tough in radio communications, which is the reason for the predominant use of dedicated hardware in current radio terminals. Thus it appears that those function chains will have similarities with today's intensive computing applications. Issues that are common in the computing world may then arise in the scope of software defined radio, in particular the mapping problem. Mapping is the assignment of system resources to applications' modules. We argue that the adopted mapping solution will have a huge impact on the performance of a SDR terminal and, therefore, propose an efficient mapping algorithm. Our approach is conceptually different from those presented so far, e.g. [6]-[9], where the minimization of the execution time is the prime objective. Since in radio communications the timing restrictions have just to be met, other types of criteria should

978-3-8007-2909-8/05/$20.00 ©2005 IEEE

be considered. The peculiarities of communication systems and the fact that they must be implemented in SDR environments, where the reconfiguration process is fairly problematic, can be summarized as:

- Most RATs access the transmission medium by means of a time-slot based division.
- Periodic execution of the same set of functions while receiving continuous or burst data streams.
- Real-time processing requirements on limited resources, where speedup is not the prime objective.
- Partial and total dynamic reconfiguration of the different layers in the protocol stack.
- An increasing heterogeneity of processing platforms.
- Highly variable computing loads and different real-time restrictions as a function of the radio standard.
- A higher efficiency in spectrum occupation generally requires more computing power.
- Higher bandwidth demands due to new user services.

The first bullet point states that the computing resource management explores some time granularity. Computing resources periodically execute the same processing chain, where the execution within a period or processing time slot must finish on time. That is, instead of speeding-up the execution, the objective is to properly deal with real-time issues.

The increasing heterogeneity of SDR platforms, that have to be managed dynamically, has a direct consequence on how to design the hardware model. Finally, more powerful computing platforms are envisaged, being the result of advanced communication standards and the rising demand on new user services. All these issues try to provide an appropriate framework for software defined radio systems, establishing the main characteristics that have to be addressed by SDR processing platforms.

II. SYSTEM MODELING

In order to perform a correct assignment of computational resources while satisfying the application's demands, information about the available and the required hardware resources have to be provided. Hence, some kind of coherent system modeling must be envisaged. We introduce a system modeling that embraces the *resource* and the *processing models*. The resource model contains all relevant information about available hardware resources as well as the platform topology or architecture. The processing model follows an equivalent approach, describing processing chains at different levels of granularity.

A. Resource Model

In the first approximation of the SDR platform we consider the processing powers of N heterogeneous devices as the major system resources. Since dealing with heterogeneous devices, the processing powers should be translated to a unique unit. MOPS (Million Operations Per Second), as proposed by Mitola in [1], could be adequate for this purpose. That is, MIPS (Million Instructions Per Second), CLBs (Configurable Logic Blocks) and maximum frequency, amongst others, typically characterizing DSPs (Digital Signal Processors), FPGAs and GPPs (General Purpose Processors), respectively, are translated to equivalent MOPS. For this contribution, we suppose that an abstraction layer with similar properties to the one proposed in [10] exists. There, the processing time is understood as an available resource that is decomposed in time slot units, which constitute the basis for the computing resources management. Equation (1) depicts the processing powers of devices P_1 to P_N after the translation to MOPS:

$$P = (P_1, P_2, ..., P_N). \tag{1}$$

Without loss of generality, we label devices by decreasing capacities. That is, we identify the device of maximum processing power as P_1 and the device of minimum processing power as P_N. In other words, $P_1 > P_2 > ... > P_N$. Note that italic letters are used for values or variables, regular letters for labels.

Apart from the processing powers, the connectivity among devices will be an equally important aspect of a SDR processing platform. It is modeled by means of a matrix that captures the bidirectional bandwidths between any two computing devices. This is the matrix B,

$$B = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1N} \\ B_{21} & B_{22} & \cdots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \cdots & B_{NN} \end{pmatrix}, \tag{2}$$

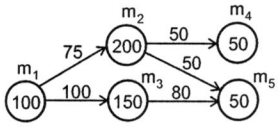
where $B_{kl} = B(k,l)$ is the offered bandwidth from device P_k to P_l, $k, l = 1, 2, .., N$, in Mega-bits per second (Mbps). B_{kk} represent the internal bandwidths of device P_i, which are considerably higher than any device-to-device link capacity. We assign B_{kk} a value of infinity for all $k = 1, 2, ..., N$. A value of zero means that there is no direct connection between the corresponding devices.

B. Processing Model

Following the same scheme as above, we characterize the software requirements: Process m_i requires m_i MOPS, $i = 1, 2, ..., M$. The bandwidth required to send data from m_i to m_j is b_{ij}, $i, j = 1, 2, ..., M$. We introduce two levels of granularity, one at *function* and another at *process levels*. At function level we identify major signal processing blocks of a SDR transmitter and/or receiver. At process level, we deal with a decomposition of these blocks. The function level is useful for the exchange of complete signal processing blocks. The process level, on the other hand, increases the mapping flexibility. Furthermore, by introducing a process level, we generalize the SDR problem, thus facilitating the adoption of already elaborated algorithms. In what follows we discuss the

granularity at process level, although all techniques can be directly applied at function level.

The set of processes m_1 to m_M may pertain to one, several or all signal processing blocks of a SDR transceiver. We use the term *task* to refer to a set of M processes. In this work, processes and their interconnections form a Directed Acyclic Graph (DAG), i.e. a graph with no directed cycle [11]. DAGs are typically used for software modeling, e.g. in [9], making the mapping problem manageable. We though allow cycles that are process-internal, or that span different processing time slots, and will consider other types of task graphs in future work. Processes are numbered according to the logical numbering scheme: If process m_i sends data to m_j, then $i < j$ [11]. One advantage of the logical numbering is that matrix b is strictly upper-triangular. Fig. 1 illustrates an example task graph and its modeling.



$m = (100, 200, 150, 50, 50)$

$$b = \begin{pmatrix} 0 & 75 & 100 & 0 & 0 \\ 0 & 0 & 0 & 50 & 50 \\ 0 & 0 & 0 & 0 & 80 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 1. Example task graph and modeling.

III. MAPPING

A. Algorithm Description

The mapping algorithm we introduce here successively maps M processes to N processing devices, starting with process m_1 and finishing with m_M. For this purpose we coin the expression *t*-mapping. It is guided by a cost function with the objective to find a low-cost solution. Different types of cost functions with different priorities may be introduced to the *t*-mapping system. In mapping step i, m_i is mapped to all N devices and the mapping costs are calculated. For $i > 1$, the mappings of previous steps are considered. These mappings are preliminary decisions rather than final mappings, because the final mapping is not obtained until finishing the step-M-processing.

Figs. 2 and 3 show a *t*-mapping example for two processors and five processes after the second and final mapping steps, respectively. The partial costs due to a fictitious cost function are given above the edges, whereas the minimum costs at each mapping decision are shown inside the nodes. A highlighted edge indicates the partial path of minimum cost among those ending at one node. We identify that in step 2 we have decided on mapping m_1 and m_2 to P_1, as well as mapping m_1 and m_2 to P_2. Both paths, i.e. momentary mappings, are maintained. Their costs are given inside the corresponding nodes below m_2. In step three of the example we, therefore, have to consider the mapping of m_1 and m_2 to P_1 when calculating the weights of the top and the upper-left-to-lower-right edges between steps two and three. The two remaining weights are obtained considering the mapping of m_1 and m_2 to P_2. This leads to the following mapping decisions: Map m_1, m_2 and m_3 to P_1 (cost = 2); Map m_1 and m_2 to P_1, m_3 to P_2 (cost = 1). As a result, one of the two paths resulting from step 2 has already been discarded. The described process is continued until obtaining the weights of all N nodes below m_M. The final mapping, indicated by circles in fig. 3, is obtained by traversing the trellis backwards along M-1 bold lines from the minimum-cost-node of step M to step 1.

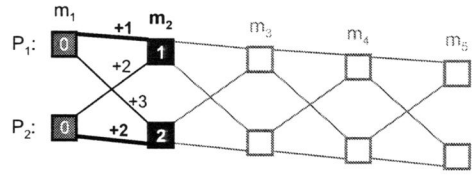


Fig. 2. *t*-mapping example after step 2.

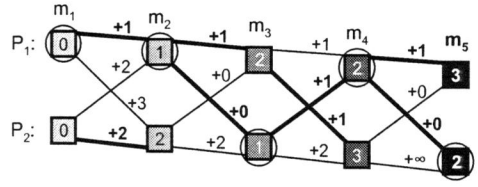


Fig. 3. *t*-mapping example after step $M = 5$.

B. Complexity Analysis

The complexity of our proposal is obtained as follows: From step 2 onwards, at each node N partial costs are calculated. There are N nodes per step and M steps in total. Hence, the computing complexity is about $M \cdot (N)^2$ times the average complexity of the cost function over the M processes, i.e.,

$$c\_t - mapping = M \cdot N^2 \cdot \overline{c\_cf} \,. \tag{3}$$

A generalization of the presented algorithm could be to not discard any path. This corresponds to a search for a, cost function specific, optimum mapping by means of computing all $(N)^M$ mappings of M processes to N devices. Equation (4) absorbs the complexity of one such mapping. Thus, the effort of computing all possible mappings is $M - 2$ orders of magnitude higher than the complexity of the *t*-mapping approach.

$$\sum_{i=1}^{M} c\_cf(m_i) = M \cdot \overline{c\_cf} \,. \tag{4}$$

IV. SIMULATIONS

A. Resource Models

Fig. 4 illustrates the five resource models we have considered for the simulations. Fig. 4a shows a *homogeneous* platform, which can be considered as a special case of its *heterogeneous* counterparts (figs. 4b and c). Figs. 4d and e depict comparable models that are based on a shared communication *bus*.

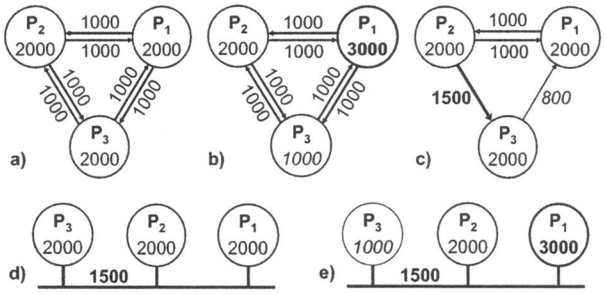


Fig. 4. Resource models.

A device with a capacity of 2000 MOPS could correspond to a GPP with a 2 GHz clock. Similarly, a DSP with a clock frequency of 250 MHz can also offer a peak performance of about 2000 MIPS or MOPS [12]. A bandwidth of about 1000 Mbps is similar to the one offered by a PCI33 interface [13].

B. Task Graphs

We have scheduled extensive simulations to evaluate the mapping proposal. In short, the *t*-mappings of a number of task graphs are compared to the correspondent best and worst-case mappings for each of the five hardware topologies. Some parameters of the task graph generation follow:

- 10000 logically numbered DAGs with M = 7, 8 or 9, uniformly distributed.
- m_i = 20, 30, ..., or 150 MOPS, uniformly distributed; i = 1, 2, ..., M.
- If there is a (no) directed arc from m_i to m_j, then b_{ij} = 50, 60, ..., or 250 Mbps, uniformly distributed (b_{ij} = 0); i = 1, 2, ..., $M-1$; $j = i+1, i+2, ..., M$.

The M processes of a task graph are randomly interconnected without violating the numbering convention. More precisely, a directed arc from process m_i to m_j, $i < j$, is drawn with a probability of 0.5, in general. This probability is downscaled for $(j-i) > (M-1)/2$, so as to lessen the chance of connecting an entry node directly to an exit node, for instance. In a complete function chain of a practical transmitter or receiver this is usually neither the case. We allow disconnected graphs but no isolated nodes [11]. A disconnected graph represents two or more parallel processing chains, and perfectly models parts of a radio transceiver that consist of independent processing paths. Examples could be RAKE fingers or WCDMA transmitter and receiver baseband processing chains [14].

The node weights pertaining to one random graph, i.e. the task's processing requirements (see fig. 1), are post-processed such that the compound processing demand is 25, 50, 75 and 85 %, respectively, of the total system capacity. These *loads* are obtained by the scaling factor *sf*:

$$load = sf \cdot \sum_{i=1}^{M} m_i \,/\, \sum_{k=1}^{N} P_k \,. \qquad (5)$$

For the second simulation series, the randomly generated task graphs are ordered in descending order of processing demands. This modifies the mapping order, but does not change a task graph's architecture. Since allocating resources for large modules, i.e. modules requiring a lot of computing resources, is generally more complicated than for smaller ones, such ordering should lead to better mapping results.

C. Cost Function

In this paragraph we introduce the cost function that we have chosen for the simulations. Although many others are possible, the one given below seems practical for the management of computing resources in SDR environments:

$$cost(k,i) = cost\_comp(k,i) + cost\_comm(k,i). \qquad (6)$$

The general term $cost(k,i)$ stands for the cost of mapping process m_i to device P_k. It is divided into the cost of computation, calculated as m_i/P_k, and the cost of communication, obtained as $b_{ji}/B(P\_m_j,P_k)$ and $b_{ij}/B(P_k,P\_m_j)$, respectively, accumulated over all $j < i$. P_m_j represents the processor that is assigned to m_j, which may be different from one mapping stage to another (see III-A). After each mapping decision, P and B are updated in such a way that the reserved processing power and bandwidths, respectively, are discounted from the corresponding value(s) prior to the decision.

D. Simulation Results

The simulation results of the first simulation series are shown in fig. 5. The left (right) bars per load represent the average relations between the costs of the minimum-cost-mappings and the costs of the *t*-mappings (maximum-cost-mappings) for the five resource models. We have computed all N^M possible mappings per task graph and resource model to obtain the mappings of minimum and maximum costs.

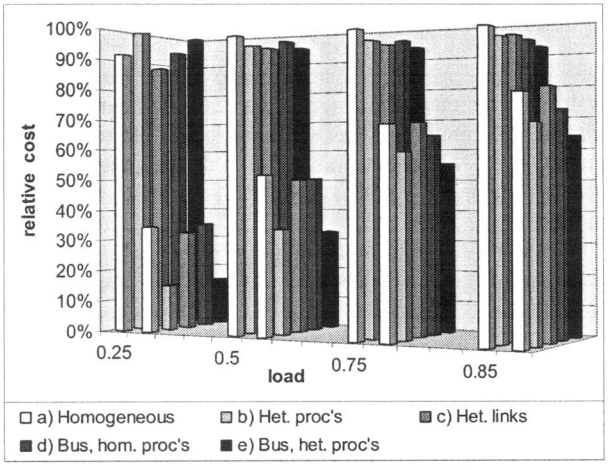


Fig. 5. Mapping results.

The results of the second simulation series are very similar to the ones presented above. In other words, fig. 5 can be considered representative for both simulations' outcomes. We thus conclude that both *t*-mapping variations achieve excellent mapping results, ranging between 90 and 100% in relation to the minimum-cost-mappings.

Nevertheless, a feasible mapping is not always possible with the *t*-mapping proposal. A feasible mapping reserves no more than 100% of any resource. Table I shows the percentage of infeasible mappings for the two *t*-mapping variations, the original or unordered (UO) and the ordered (ORD) cases. In line with our expectations, we observe that the ORD-*t*-mapping leads to better numbers in this aspect. Table I also reveals that for the third resource model, the *t*-mappings fail for a number of task graphs, which is the more pronounced the higher the processing load. Task graphs leading to infeasible *t*-mappings are not incorporated in fig. 5.

TABLE I. INFEASIBLE T-MAPPINGS IN %.

Resource models\ load		0.25	0.5	0.75	0.85
a) Homogeneous	UO	0	0.01	0.61	13.7
	ORD	**0**	**0**	**0**	**0.19**
b) Heterogeneous processors	UO	0	0	0.82	11.7
	ORD	**0**	**0**	**0**	**0.06**
c) Heterogeneous links	UO	0	0.25	28.6	60.5
	ORD	**0**	**1.35**	**11.0**	**32.9**
d) Bus, homogeneous processors	UO	0	0.07	2.7	18.0
	ORD	**0**	**0.19**	**1.5**	**3.4**
e) Bus, heterogeneous processors	UO	0	0	1.04	13.6
	ORD	**0**	**0**	**0.2**	**2.2**

Table II presents the average number of feasible mappings for the different resource models and loads. Among other things it shows that the average number of feasible mapping possibilities for *resource model c* is about one order of magnitude lower than for any of the other four models. Therefore, both *t*-mapping variations have difficulties to successfully map task graphs to this model.

TABLE II. AVERAGE NUMBER OF FEASIBLE MAPPINGS.

Resource models\ load	0.25	0.5	0.75	0.85
a) Homogeneous	8436	7495	2151	677
b) Heterogeneous processors	8123	4093	991	351
c) Heterogeneous links	709	519	75	21
d) Bus, homogeneous processors	5851	4999	1242	381
e) Bus, heterogeneous processors	5567	2781	685	236

V. CONCLUSION

In the present article we have expressed the need for mapping in software defined radio and established a system model that encompasses the available computing resources and the processing chain requirements. We have, further, introduced a simple mapping algorithm that requires very few computational resources and, at the same time, obtains close to optimum results. This implies that it is very likely that such a proposal will have the capability to ensure the real time operation of many different function chains. Although the scheduled simulations are on a general basis, the entire framework is envisaged for future SDR scenarios. We conclude this work, emphasizing that the framework is easily extensible and open for many different cost functions.

ACKNOWLEDGMENT

This work has been supported by CYCIT (Spanish National Science Council) under grant TIC2003-08609, which is partially financed from the European Community through the FEDER program.

REFERENCES

[1] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, vol. 33, no. 5, pp. 26–38, May 1995.

[2] E. Buracchini, "The software radio concept," *IEEE Commun. Mag.*, vol.38, iss.9, pp.138–143, Sept. 2000.

[3] Xilinx homepage, www.xilinx.com

[4] M. Cummings, S. Haruyama, "FPGA in the software radio," *IEEE Commun. Mag.*, vol. 37, iss. 2, pp. 108–112, Feb. 1999.

[5] A. Gathener, T. Stetzler, M. McMahan, E. Auslander, "DSP-based architectures for mobile communications: past, present and future," *IEEE Commun. Mag.*, vol. 38, iss. 1, pp. 84–90, Jan. 2000.

[6] A. H. Alhusaini, V. K. Prasanna, C. S. Raghavendra, "A framework for mapping with resource co-allocation in heterogeneous computing systems," in *Proc. 9th IEEE Heterogeneous Computing Workshop (HCW 2000)*, Cancun, Mexico, May 2000, pp. 273–286.

[7] J.-K. Kim, et al., "Collective value of QoS: A performance measure framework for distributed heterogeneous networks," in *Proc. 15th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS 2001)*, San Francisco, CA, April 2001, pp.810–823.

[8] M. Tan, H. J. Siegel, J. K. Antonio, Y. A. Li, "Minimizing the application execution time through scheduling of subtasks and communication traffic in a heterogeneous computing system," *IEEE Trans. Parallel Distrib. Systems*, vol. 8, iss. 8, pp. 857–871, Aug. 1997.

[9] A.-R. Rhiemeier, F. Jondral, "Mathematical modeling of the software radio design problem," *IEICE Trans. Commun.*, vol. E86-B, no. 12, Dec. 2003.

[10] X. Revés, V. Marojevic, A. Gelonch, "Run-time and development framework for heterogeneous hardware radios," in *Proc. 1st IST-E2R Workshop Reconfigurable Mobile Systems and Networks Beyond 3G*, Barcelona, Spain, Sept. 2004.

[11] D. F. Robinson, L. R. Foulds, *Digraphs: Theory and Techniques*. Gordon and Breach Science Publisher Inc., 1980.

[12] *SPRU395B: TMS320C64x Technical Overview*, Texas Instruments, Dallas, TX, Jan. 2001.

[13] *PCI Local Bus Specification, Revision 2.2*, PCI Special Interest Group, Portland, OR, Dec. 1998.

[14] R. Tanner, J. Woodard, *WCDMA – Requirements and Practical Design*. John Wiley and Sons Inc., 2004.