The FlexCRM Project

Vuk Marojevic, Ismael Gomez, Antoni Gelonch

Universitat Politècnica de Catalunya

July 2011

Contents

This document explains the ALOE computing resource management approach. It presents the motivation for computing resource management, the organization of the ALOE management framework, the time management mechanisms, the system modeling, and the computing resource management. We indicate our current computing resource management research and present a list of further readings.

1 Motivation

SDR presents a hard real-time computing challenge. Taking into account the evolution of wireless systems it is clearly stated that whereas spectrum efficiency increases lineally its requirements in computing resources increases exponentially. Therefore, the flexibility, based on the reconfiguration capacity of terminals and network equipment, is based on capability of their reconfiguration managers, which need to track the states of the computing resources.

An SDR processing chain, SDR application or waveform, is the part of an SDR transceiver that is implemented in software. It can be understood as a set of concurrent processes that continuously process and propagate real-time data. Such a processing chain is not specifically tailored but rather executable on any general-purpose platform with sufficient computing capacity. Therefore, an automatic mapping process needs to dynamically assign software modules to hardware resources, while meeting all computing system constraints. Wireless and SDR systems reveal specific aspects, essentially regarding flexibility and efficiency, that should be jointly considered:

1. Time slot based division of the transmission medium (radio time slot);
2. Continuous data transmission and reception;
3. RAT-specific quality of service (QoS) targets;
4. Real-time computing requirements and limited computing resources;
5. Different constraints and computing loads for different RATs and radio conditions;
6. Dynamic reconfiguration of the protocol stack, either partial or total;
7. Heterogeneous multiprocessor execution platforms.

2 Organization

Our computing resource management proposal consists of several modules. We distinguish between the *computing system modeling* and the *computing management mechanisms*. The basis for the resource management is the time management of Section 3.

The computing system modeling (Section 4) creates suitable models of SDR platforms and SDR applications, capturing the available and required computing resources.

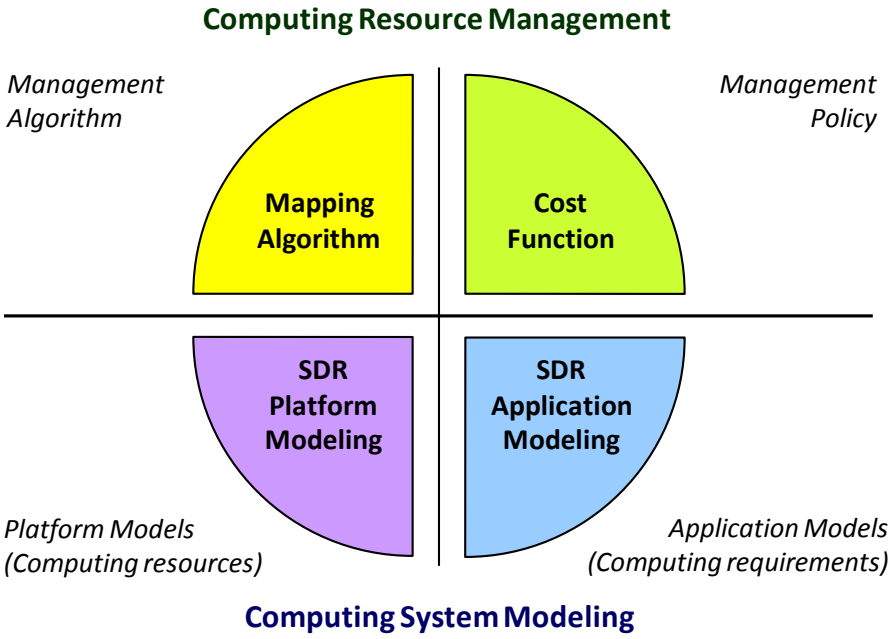

Figure 1. FlexCRM organization.

The computing resource management (Section 5) is based on the use of general-purpose mapping algorithms and specific cost functions. It manages (allocates, keeps track of, and updates) the available computing resources as a function of the computing requirements and management policies.

3 Time Management

3.1 Metrics

An SDR platform represents an SDR mobile terminal or an SDR network element. These platforms comprise a few or many heterogeneous processing devices, such as FPGAs, DSPs, and GPPs, which communicate with each other. An FPGA's prime resource is the logic area for parallel processing, which can be converted to multiply-accumulate operations (MACs) per time unit when using well-defined benchmarks (filter, FFT, and so forth). DSP, GPP, and MP-SoC performances are typically given in million instructions per second (MIPS).

The processing powers and the inter-processor bandwidths are the primary resources of SDR platforms. We consider million operations per second (MOPS) as the basic unit for characterizing the processing powers and mega-bits per second (Mbps) for the (inter-)processor

communication capacities. We correspondingly apply the same metrics for capturing SDR applications' processing and data flow requirements. The implicit timing requirements need to be specified as a function of the radio link timing requirements.

3.2 Time-Slot Division and Pipelining

We consider processing time as just another limited computing resource. MOPS and Mbps embed this critical resource and thus permit an implicit time management. In continuation we discuss two mechanisms that ease the computing resource management (Section 5).

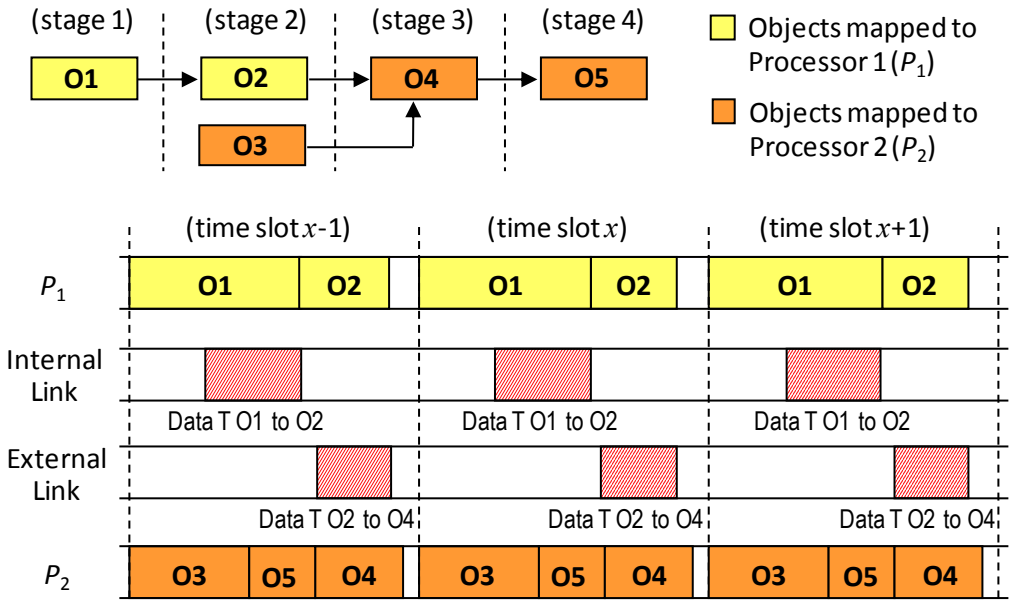

Figure 2. Time slots and pipelining.

Data that is transmitted or received over the wireless link needs to be processed for as long as there is data to transmit or receive. An SDR application will execute during the entire user session or until it is exchanged by another one. We thus propose breaking up the continuous execution into periodic executions by dividing the computing resource time in equidistant computing time slots and the SDR application in pipelining stages. The following figure illustrates this.

The pipelined execution of an SDR application establishes that, in any time slot, all SDR functions process and propagate some part of the data. That is, the same processing and data transfers repeat each time slot on a different data portion (Figure 2). We can then derive the new units million operations per time slot (MOPTS) and mega-bits per time slot (MBPTS) as $t \cdot$ MOPS and $t \cdot$ Mbps, where t is the time slot duration that is specified as a function of the latency requirements and the number of pipelining stages (Section 3.3). MOPTS and MBPTS synchronize the available computing resources with the time slot management and are the basic units for the SDR (Section 4).

3.3 Scheduling: Meeting the Real-Time Computing Constraints

The adopted computing resource management facilities and our computing system modeling permit mapping an SDR application to an SDR platform on time slot basis.

We assume that coprocessors facilitate the concurrent data processing and data propagation on all processor's in- and outputs. Since repetitive operations on data samples and continuous outputs, often one per execution cycle, characterize digital signal processing, we may further assume that the software and hardware facilitate the immediate propagation of processed data samples. The SDR framework finally needs to manage the synchronized execution on all processors and provide pipelining and buffering mechanisms, among others, for the proper and timely data delivery. ALOE provides these mechanisms.

The usually complex scheduling process can, on the basis of a feasible mapping - a mapping that reserves no more than 100 % of any available computing resource - and under the above assumptions, be simplified to N independent local scheduling tasks. Particularly, a processor's local scheduler is capable of organizing the execution sequence of the corresponding SDR functions' portions and their data transfers within the given time-slot boundaries. This ensures that the input data of any SDR application's module or set of modules is processed according to its arrival rate so that no data is accumulated anywhere in the processing chain, meeting the minimum bit rate requirement.

The time slot duration t times the number of the application's time slots n is the pipelining latency in case of a feasible schedule on each processor. We specify t as

$$t = L / n \text{ [SPTS]}$$

to meet the application's maximum allowable latency L, where SPTS stands for seconds per time slot. We assume that t is large enough for the (efficient) execution of any SDR function in the processing chain.

4 Computing System Modeling

The SDR computing system modeling consists of the platform modeling and the application modeling. The platform modeling characterizes SDR platforms and their computing resources, whereas the application modeling abstracts SDR applications and their computing requirements. We identify four relevant types of computing resources: processing, bandwidth, memory and energy resources. The computing requirements correspondingly include the processing, dataflow, memory and energy demands. The following resource models and requirements address processing capacities/requirements and interpreocessor bandwidth capacities/ requirements. These models are the basis for a correct computing resource management (Section 5).

In order to simplify the real-time computing resource management, time is considered as just another computing resource, which is implicitly modeled and managed (see also Section 3).

4.1 Platform Modeling (Computing Resources)

We model an SDR platfrom as N interconnected processors P_1, P_2, ..., P_N. The *device model* $\boldsymbol{C}$ is an N-element vector that captures their processing capacities:

$$\boldsymbol{C} = (C_1, C_2, \ldots, C_N) \text{ [MOPTS]}. \quad (1)$$

The *communication model*, N times N matrix $\boldsymbol{Bx}$, indicates the (inter)processor bandwidth capacities, where B_{xy} indicates the available bandwidth for moving data from processor P_x to processor P_y:

$$\boldsymbol{Bx} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1N} \\ B_{21} & B_{22} & \cdots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \cdots & B_{NN} \end{pmatrix} = \begin{pmatrix} \infty & B_{12} & \cdots & B_{1N} \\ B_{21} & \infty & \cdots & B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N1} & B_{N2} & \cdots & \infty \end{pmatrix} \text{ [MBPTS]}. \quad (2)$$

$\boldsymbol{Bx}$ informs about the communication topology (interconnectivity network) and the communication resources (bandwidths), assuming a network that consists of unidirectional communication links between each pair of processors. Additional information is needed for modeling shared links. We therefore suggest a more general communication modeling that distinguishes between the communication topology (3) and the communication resources (4).

$$\boldsymbol{I} = \begin{pmatrix} I_{11} & I_{12} & \cdots & I_{1N} \\ I_{21} & I_{22} & \cdots & I_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_{N1} & I_{N2} & \cdots & I_{NN} \end{pmatrix} \quad (3)$$

represents the logical interconnection model, where $I_{uv} \in 1, 2, \ldots, N \cdot N$ is a unique numerical label of the logical link between P_u and P_v. A logical link corresponds to a directed (unidirectional) communication line between a pair of processors. These logical links map to physical link bandwidths

$$\boldsymbol{B} = (B_1, B_2, \ldots, B_N, B_{N+1}, \ldots, B_{N \cdot N}) = (\infty, \infty, \ldots, \infty, B_{N+1}, \ldots, B_{N \cdot N}) \text{ [MBPTS]}. \quad (4)$$

B_u, where $u = I_{32}$ for instance, is the maximum bandwidth that is available for the directed data transfer from the local data memory of processor P_3 to the local data memory of processor P_2. It would be zero if the physical link is unavailable or nonexistent. The first N elements of $\boldsymbol{B}$, B_1 to B_N, capture the processor-internal communication resources of processor P_1 to P_N; hence, $I_{uu} \in 1, 2, \ldots, N$. Since processor-internal data movements are typically orders of magnitude faster than processor-external data transfers, we can label logical links so that $B_1 \geq B_2 \geq B_3 \geq \ldots \geq B_{N \cdot N}$. Unused elements in (4) are filled with 0s.

4.2 Application Modeling (Computing Requirements)

We model an SDR application or waveform as a signal processing chain that consists of the M SDR functions $f_1, f_2, ..., f_M$.

The *function model* $\boldsymbol{c}$, an M-ary vector, provides their processing requirements, whereas the *dataflow model* $\boldsymbol{b}$, an M-times-M matrix, indicates the precedence constraints between SDR functions and provides the data flow (bandwidth) requirements:

$$\boldsymbol{c} = (c_1, c_2, ..., c_M) \text{ [MOPTS]}, \tag{5}$$

$$\boldsymbol{b} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \cdots & b_{MM} \end{pmatrix} \begin{pmatrix} 0 & b_{12} & \cdots & b_{1M} \\ 0 & 0 & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \text{ [MBPTS]}. \tag{6}$$

The *stage model* $\boldsymbol{s}$ finally specifies the pipelining stages associated with SDR functions f_1 through f_M [2]:

$$\boldsymbol{s} = (s_1, s_2, ..., s_M). \tag{7}$$

5 Computing Resource Management

The computing resource management relies on our computing system modeling described in Section 4. The figure below shows the interactions.

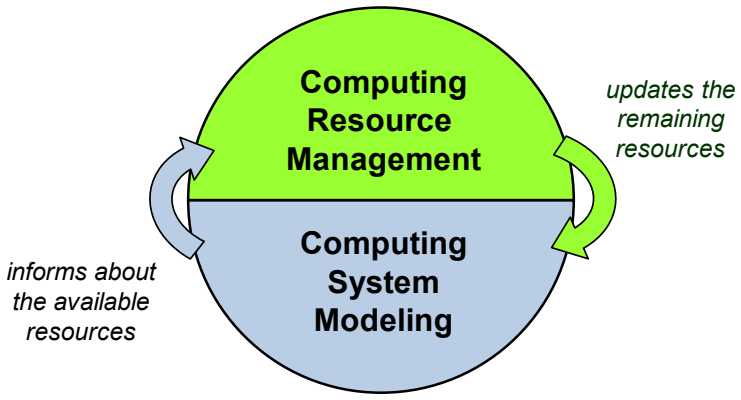

Figure 3. Management interactions.

5.1 The t_w-mapping

The t_w-mapping is a general-purpose mapping algorithm. It is a windowed dynamic programming algorithm, where w indicates the window size. The mapping process is organized by the t_w-mapping diagram, which contains a trellis of N times M (row times column) t-nodes. A t-node is identified as $\{P_j, f_i\}$ and absorbs the mapping of SDR function f_i to processor P_j. Any t-

node at *step i* (column *i* in the t_w-mapping diagram) connects to all *t*-nodes at *step i*+1. The sequence of processors $[P_{k(0)}\ P_{k(1)}\ \ldots\ P_{k(w)}]_i$ identifies the *w*-path, a path of length *w*, that is associated with *t*-node $\{P_{k(1)}, f_i\}$, where $P_{k(0)}$ is the *w*-path's origin processor at *step i*–1 and $P_{k(w)}$ the destination processor at *step i*+*w*–1.

The main feature of the t_w-mapping is that it is cost function independent. That is, any cost function can, in principle, be applied. The cost function guides the mapping process. It is responsible for managing a platform's available computing resources and an application's real-time processing requirements.

The algorithm sequentially pre-assigns, or pre-maps, processes to processors, starting with SDR function f_1 and finishing with SDR function f_M (parts I and II of the algorithm). This is followed by a post processing that determines the final mapping (part III).

t_w-mapping, part I

Part I consists of pre-mapping SDR function f_1 to all *N* processors and storing the pre-mapping costs at *t*-nodes $\{P_1, f_1\}$ through $\{P_N, f_1\}$. Costs are computed due to some cost function (Section 5.2).

t_w-mapping, part II

At *step i* of part II ($2 \le i \le M–w+1$) the t_w-mapping examines all N^w *w*-paths that are associated with $\{P_{k(1)}, f_i\}$. These *w*-paths originate at a *t*-node at *step i*–1, pass through $\{P_{k(1)}, f_i\}$, and terminate at a *t*-node at *step i*+*w*–1. The following figure illustrates this for $P_{k(1)} = P_1$.

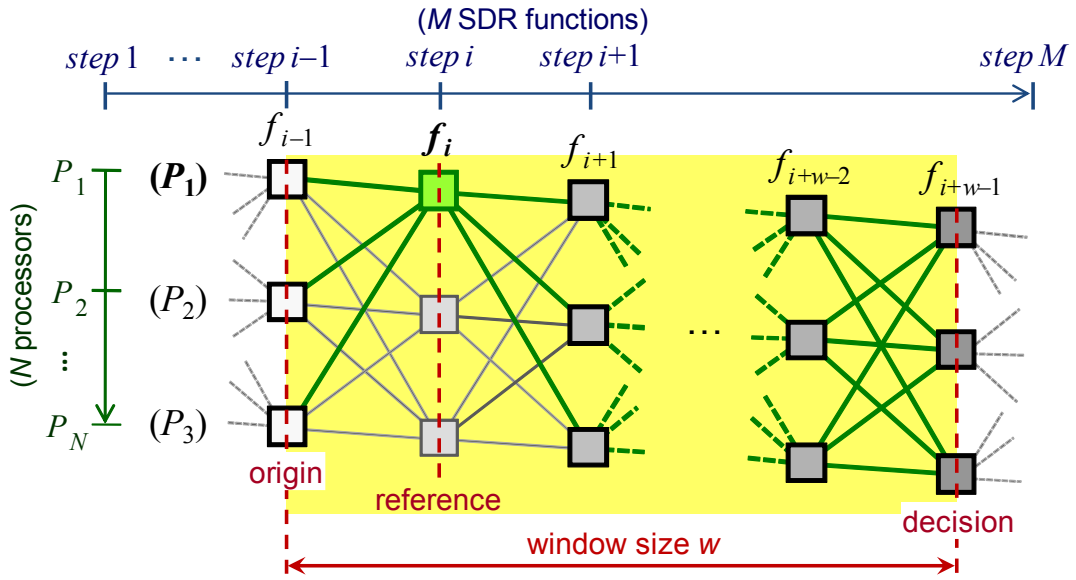

Figure 4. t_w-mapping diagram and examined paths at *t*-node $\{P_1, f_i\}$.

In case that $i < M–w+1$, the algorithm highlights the edge between a *t*-node at *step* $i–1$ and *t*-node $\{P_{k(1)}, f_i\}$ that corresponds to the minimum-cost *w*-path. The minimum-cost *w*-path is the path that is associated with the minimum accumulated cost due to the corresponding pre-mappings of f_1, f_2, ..., and $f_{i+w–1}$, where the *w*-path's origin *t*-node provides the pre-mapping information of f_1 to $f_{i–1}$. The algorithm then stores the cost and the remaining resources up to *t*-node $\{P_{k(1)}, f_i\}$ at $\{P_{k(1)}, f_i\}$. It (simultaneously) processes all *t*-nodes at *step i* before considering those at *step* $i+1$.

If $i = M–w+1$, however, the complete *w*-path of minimum cost is highlighted. The total cost and the remaining resources are then stored at $\{P_{k(1)}, f_{M–w+1}\}$. After having processed all *N* *t*-nodes at step $M–w+1$, part III of the algorithm starts.

t_w-mapping, part III

Part III tracks the t_w-mapping diagram back- and forward along the highlighted edges, starting at the minimum-cost *t*-node at *step* $M–w+1$. This process finds the complete mapping solution for the given problem and cost function.

5.2 Cost Function

The cost function is responsible for managing the available computing resources of SDR platforms while allocating the required resources to SDR applications. We generally define it as the sum of weighted *cost terms*, where each *cost term* captures the relation between the required and available resource of a specific type. Each of these terms must be less than or equal to 1. Otherwise, we would reserve more resources than available. Hence, the cost function computes the cost of a pre-mapping as a function of the available and the required resources. This implies dynamic resource updates as indicated by Figure 3.

For managing the processing and interprocessor bandwidth resources we define the cost function as

$$cost = q \cdot cost\_comp + (1–q) \cdot cost\_comm. \quad (8)$$

This two-term cost function manages the available processing and bandwidth resources, while trying to meet the corresponding computing resource requirements. Weight q is defined in interval [0, 1]. It defines the relative importance of the *computation cost* with to the *communication cost*.

The static cost function weight q has been shown to be significant in different case studies [2]. Its optimum value depends on the mapping problem, that is, the platform architecture and computing resources as well as the SDR application characteristics. As a result, we introduce another set of weights, k_1 and k_2, which are automatically assigned as a function of the mapping problem. Equation (8) then becomes

$$cost = q \cdot k_1 \cdot cost\_comp + (1–q) \cdot k_2 \cdot cost\_comm. \quad (9)$$

The two new weights can be statically or dynamically assigned. They represent the ratio between the total computing requirements and the total computing resources of each kind. In the case of a dynamic assignment, k_1 and k_2 are dynamically updated as a function of the computing requirements still to be mapped and the remaining computing resources. This update occurs at each mapping step i (Figure 4), for each individual t-node, whereas no updates are performed throughout the processing of the associated w-paths.

6 Current Research

We currently investigate how to minimize the pipelining latency at the mapping and scheduling stages. Therefore, we examine new cost functions and evaluate their performance in different computing resource management scenarios. We, furthermore, address the scalability of the t_w-mapping with the objective of applying it to large processor arrays and multiprocessor systems-on-chip (MP-SoCs). We work on extensions of the t_w-mapping algorithm itself, but, also, on pre and postprocessing add-ons.

Further Readings

The following documents contain further information about the ALOE computing resource management approach:

[1] V. Marojevic, X. Revés, A. Gelonch, "A computing resource management framework for software-defined radios," *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1399-1412, Oct. 2008. Online available: http://www.tsc.upc.edu/grcm/images/stories/IEEE_Trans._Computers_marojevic_2008.pdf

[2] V. Marojevic, "Computing resource management in software-defined and cognitive radios," Ph.D. Dissertation, Universitat Politècica de Catalunya (UPC), Barcelona, July 2009. Online available: http://flexnets.upc.edu/trac/wiki/Publications