ALOE Session 8: Computing Resource Management Tools

Vuk Marojevic, Ismael Gomez, Antoni Gelonch
Universitat Politècnica de Catalunya

July 2011

1. Objective

In this session you will experience a set of computing resource management tools. These tools help understanding and analyzing ALOE's computing resource management decisions.

2. Overview

- Introduction
- ALOE's computing resource management framework
- Computing resource management tools

3. Requirements

- Windows, 32 bits (send an email to flexnets.pmt@upc.edu if you are using another system)
- Matlab
- Basic Matlab programming skills

4. Versions

We continuously evolve the ALOE framework and tools. The latest version of the computing resource management framework and tools can be downloaded from http://flexnets.upc.edu/trac/wiki/ResourceManagement. This session does not require the download and installation of ALOE or the computing resource management framework. The MEX-file, which is used in this session, was generated for 32 bits Windows. Send an email to flexnets.pmt@upc.edu if you are using another system.

5. Procedure

5.1. Introduction

Software defined radio (SDR) represents a distributed real-time computing challenge. The emerging wireless standards with different operating modes, increasing processing requirements, and radio link dependencies bring along new resource management implications and opportunities.

The basic concept of SDR refers to software implementations of the signal processing modules that have traditionally been implemented in hardware (e.g. mixers, filters, modulators). SDR thus leverages flexibility, enabling the easy modification or changes of filter characteristics, modulation schemes, or other transmission parameters. Modifying or replacing some parts of the software can completely change the radio functionality of the transmitter, or receiver, or both. This allows improving the system performance without redesigning the hardware. SDR is thus well suitable for modern wireless communications, with coexisting radio standards, varying channel conditions and user demands.

The tremendous complexity increase of each new generation of wireless communications, on the other hand, calls for computing efficient implementations, rather than software solutions. Continuous advances in processing technology, power efficient multi-core devices and very fast device-to-device communication fabric will eventually lead to SDR base stations and mobile terminals. The multiprocessing execution environment increases the processing power through parallel processing units. The strict timing constraints of wireless communications services then require efficient multiprocessor mapping and scheduling solutions that can allocate the distributed computing resources in real-time while meeting all computing system constraints. The computing resource manager should be able to respond to changes in the radio environment, enabling seamless reconfiguration and cognitive radio.

5.2. ALOE's computing resource management framework

ALOE's computing resource management framework distinguishes between the computing system modeling and the computing resource management [1] [2] [3]:

- The computing system modeling creates suitable models of SDR platforms and applications. It currently captures the processing and data flow resources and requirements while implicitly accounting for the waveform's timing constraints.
- The computing resource management part consists of several mapping algorithms and a parametric cost function.

The modeling is based on the pipelined execution principle, where data is processed and propagated in blocks. ALOE synchronizes the distributed signal processing and provides interprocessor communication and buffering mechanisms.

The mapping process is defined by a general-purpose algorithm and guided by the cost function. This way the application modules (digital signal processing blocks) are distributing among the limited and distributed computing resources (multiprocessor execution platform). Three algorithms are implemented:

- t_w-mapping: The t_w-mapping is based on the dynamic programming principle, where w indicates the decision window size [1]. You will learn more about the t_w-mapping in this session.

- g_w-mapping: The g_w-mapping is an extended or windowed greedy algorithm [1]. Its computing complexity depends linearly on the problem size for $w = 1$. This makes the g_w-mapping applicable for large-scale computing systems. For simpler problems, it may be used as the baseline algorithm.

- *opt*-mapping: The *opt*-mapping does an exhaustive search over the entire solution space for optimally solving a mapping problem with respect to the given cost function. The *opt*-mapping corresponds to the t_w-mapping with the maximum window size of $w = M–1$, M being the number of application modules.

The cost function manages the available computing resources, processing powers and interprocessor communication bandwidths. It is fully described in [3].

ALOE's computing resource management (CRM) framework is completely implemented in C. The CRM API allows an easy integration of new algorithms and cost functions. The CRM package, available for free download from http://flexnets.upc.edu/trac/wiki/ResourceManagement, contains this API and a simulation test suite with different use case scenarios. That is, different SDR application and platform models are readily available for use. Some of these are experienced in ALOE Session 7 [2].

5.3. The Matlab interface

The CRM framework is implemented as a library and linked with Matlab through a binary MEX-file. A binary MEX-file is a Matlab subroutine that is dynamically linked to a C or C++ subroutine. This permits accessing C/C++ functions from Matlab scripts. Because programs are compiled using a C/C++ compiler for a specific processor, their execution is typically much faster than the execution of equivalent Matlab routines.

The Matlab interface permits creating SDR application and platform models in Matlab and passing them to the computing resource management framework, which returns the mapping. The interface, more precisely, accepts custom computing system models and mapping parameters—specified in an m-file—and returns the mapping solution, cost, and other CRM output parameters. This is convenient for rapidly creating new waveform and platform models and testing the algorithm performances without the need for modifying and recompiling the CRM framework.

Download the tools from http://flexnets.upc.edu/trac/wiki/ResourceManagement. Extract (unzip) the files, which are listed in Table 3 in the appendix. Open Matlab and switch to the path where you extracted the files. Take a look at *aloe8a.m*, which defines an example computing system model. This model corresponds to the custom model of ALOE Session 7, Section 5.4 [2] or [1, Fig. 4.5 on page 56]. Figure 1 below takes a snapshot. Take a look at how the matrices are defined. Beware that Matlab indexes array elements different than C: The first index is '1' in Matlab and '0' in C.

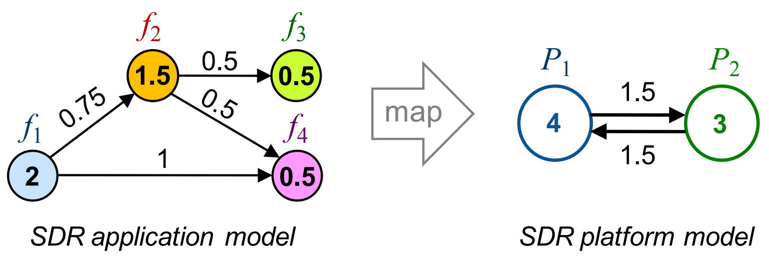


Figure 1 - Example computing system model.

Table 1 - Matlab interface parameters.

Parameter	Range	Description
C	Real, nonnegative	Processing powers (N-ary row vector)
I	1, 2, …, N·N	Interprocessor communication network topology (NxN matrix)
B	Real, nonnegative	Interprocessor bandwidth resources (N·N-ary row vector)
c	Real, nonnegative	Processing requirements (M-ary row vector)
b	Real, nonnegative	Data flow demands (MxM matrix)
alg	0, 1, 2	Algorithm selector: 0 = g_w-mapping, 1 = tw-mapping, 2 = opt-mapping
w	1, 2, …, M–1	Window size
q	[0, 1]	Static cost function weight
k	0, 1, 2	Dynamic cost function weight (k-weighting) selector: 0=disabled; 1=static; 2=dynamic [3]
order	0, 1, 2	Mapping order: 0 = original order; 1 = by decreasing processing requirements; 2 = by decreasing bandwidth requirements
mhop	0, 1, 2	Multihop enabler: 0 = direct communication (single hops) only; 1 = 2-hops if and only if necessary; 2 = single and 2-hops, whichever is less costly

The *aloe8a* script calls the *mapper_matlab.mexw32* file with 11 input parameters (line 40 in *aloe8a.m*). The first 5 parameters define the computing system model. The modeling vectors and matrices *C*, *I*, *B*, *c*, and *b* are discussed in ALOE Session 7 [2], among others [1] [3]. The remaining 6 parameters specify the mapping algorithm and cost function parameters. All input parameters are summarized in Table 1.

Try executing the script. Therefore, type

```
aloe8a
```

in the Matlab command window. Figure 2 shows the output.

Figure 2 - Mapping output.

```
>> aloe8a
Using Mx=4, Nx=2, alg=1, w=2, q=0.5000, k-weighting=0, order=0, hop=0

cost =

    1.2083


result =

           0
           1
           1
           0

Do you want to execute trellis example? (y/n) n
>>
```

The *result* vector contains the mapping information as returned from the CRM library. It contains 4 entries, result[1] to result [4]. "*result* = 0 1 1 0" indicates the following mapping: SDR functions f_1 and f_4 to processor P_1 (result[1] = result[4] = 0), and f_2 and f_3 to the processor P_2 (result[2] = result[3] = 1). The cost of this mapping solution is 1.2083. Fig. 4.18 on page 59 of [1] shows the same result with a cost scaled by $q^{-1} = 2$ (because no q weighting was applied there).

Respond with `n` to the question

```
Do you want to execute trellis example? (y/n)
```

and press enter.

Edit *aloe8a.m* for executing another mapping algorithm (line 32) or window size (line 33). Compare the outcomes with those of [1, pages 57-62].

5.4. The tw-mapping algorithm

The t_w-mapping is a dynamic programming algorithm. A dynamic programming algorithm solves an optimization problem by computing cost metrics first and then unrolling the problem backwards, based on previously computed metrics [4].

Here, the metrics are computed using the implemented cost function [2] [3]. The comparison of pre-mapping costs lead to pre-mapping decisions, where one out of *N* edges is selected at each *t*-node. (*N* is the number of processors.) For $w > 1$, more SDR functions are considered for the cost calculation and comparison (Figure 3), which leads to a single edge selection, except when processing the *w*-paths that reach the final *t*-nodes. This essentially describes part II of the t_w-mapping process [1] [2]. It corresponds to the forward processing part of the dynamic programming approach.

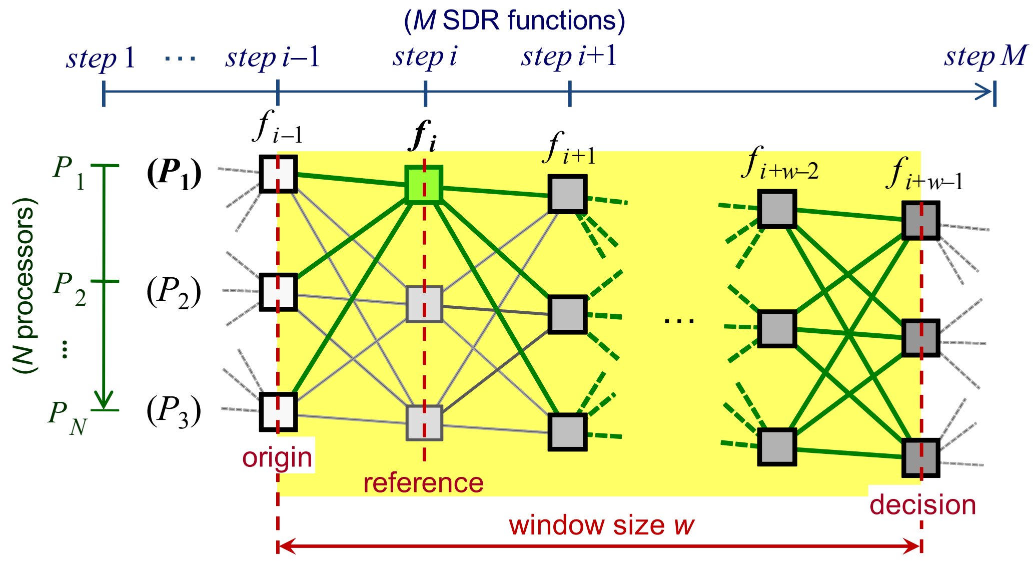


Figure 3 - tw-mapping diagram and examined paths at *t*-node {P_1, f_i}.

Part III of the t_w-mapping first selects the *t*-node of minimum cost at step $M–w+1$ (assuming that we number the SDR function and mapping steps from 1 to *M*, as indicated in Figure 3). We may call this *t*-node $\{P^*, f_{M–w+1}\}$. This node defines the processor assignment for SDR function $f_{M–w+1}$: $f_{M–w+1} \rightarrow P^*$. Moving along the highlighted edges from $\{P^*, f_{M–w+1}\}$ and until reaching a node at step 1 and step *M* provides the entire mapping solution. More precisely, for $w > 1$, the *w*-path associated with the minimum cost *t*-node at step $M–w+1$ defines the mapping of the last $w–1$ SDR functions. The remaining nodes are obtained by backtracking the trellis from $\{P^*, f_{M–w+1}\}$ along the highlighted edges until reaching a *t*-node at step 1. The traversed nodes represent the t_w-mapping solution for the given problem and cost function.

Run the *aloe8a* script again, this time answering with yes (y) to the question that appears after the *result* vector. Therefore, type `y` and press enter. The result should look like Figure 4, where the green nodes represent the mapping solution given in digital form in Figure 2. (Note that the trellis nodes are numbered from top to bottom and from left to right, just as in Figure 3. Also bear in mind that the digital mapping representation uses indexes from 0 to $N–1$, as opposed to the visualization tools, which index processors from 1 to *N*.)

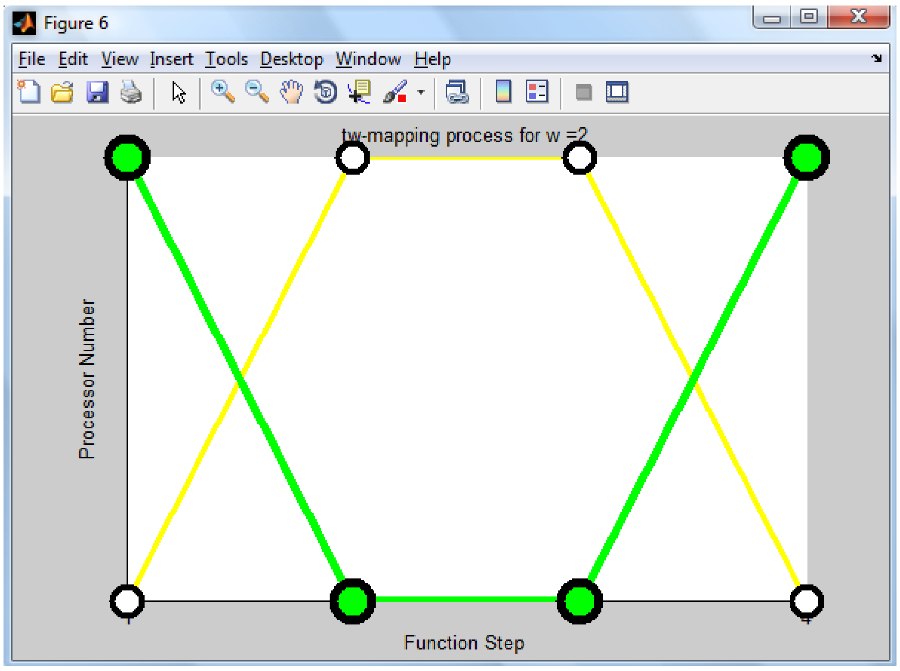


Figure 6
File Edit View Insert Tools Desktop Window Help
tw-mapping process for w =2
Processor Number
Function Step

Figure 4 - The t_2-mapping process illustration.

The animation illustrates the pre-mapping decisions, which are based on the cost calculations and comparisons of the different mapping options. That is, at each *t*-node N^w mapping options are computed and compared leading to a unique decision, which is visualized. (Reference [1, chapter 4] contains the details about the cost calculation-comparison-decision cycle.) Part III of the t_w-mapping is shown in green lines.

You may understand the relation between t_w-mapping part II and III better if you see both results in parallel. Therefore, after executing *aloe8a* and following the t_w-mapping process animation, you may execute the *trellis2* function with three input parameters:

```
>> trellis2(edges,w,cost_min)
```

This will take the previously obtained mapping outcome information and redo the t_w-mapping process animation of part II.

Edit *aloe8a.m*, choosing the following mapping parameters (lines 32 to 37):

- `alg=1;`
- `w=1;`
- `q=0.5;`
- `k=0;`
- `order=0;`
- `mhop=0;`

Save *aloe8a.m* and execute it. Answer with `y` to the question appearing on the Command Window and press enter. After following the t_w-mapping animation process execute `trellis2(edges,w,cost_min)` from the Command Window. You should observe the plots shown in Figures 5 and 6.

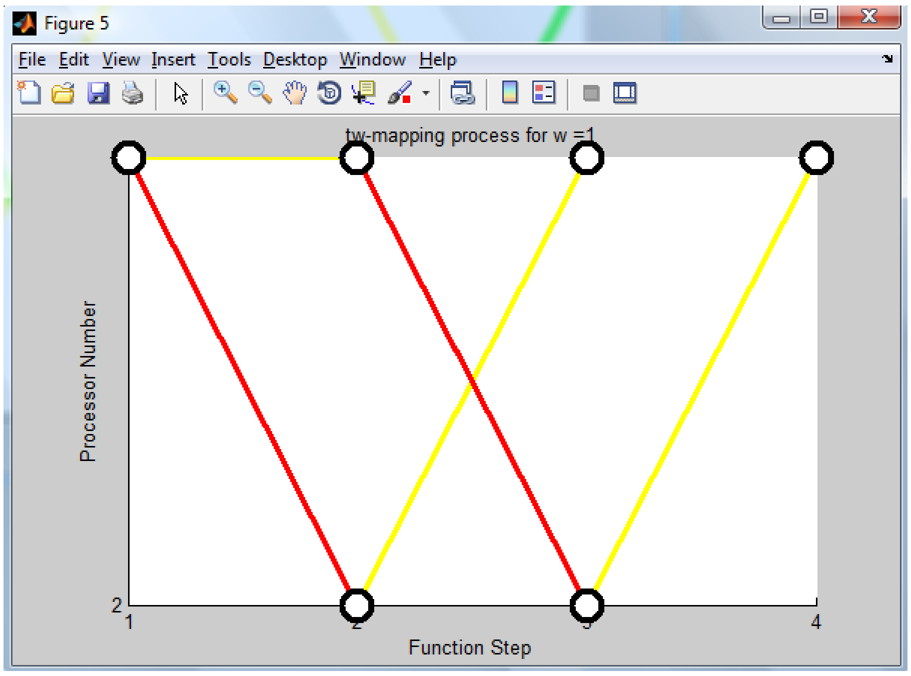


Figure 5 - t_1-mapping, part II process illustration.

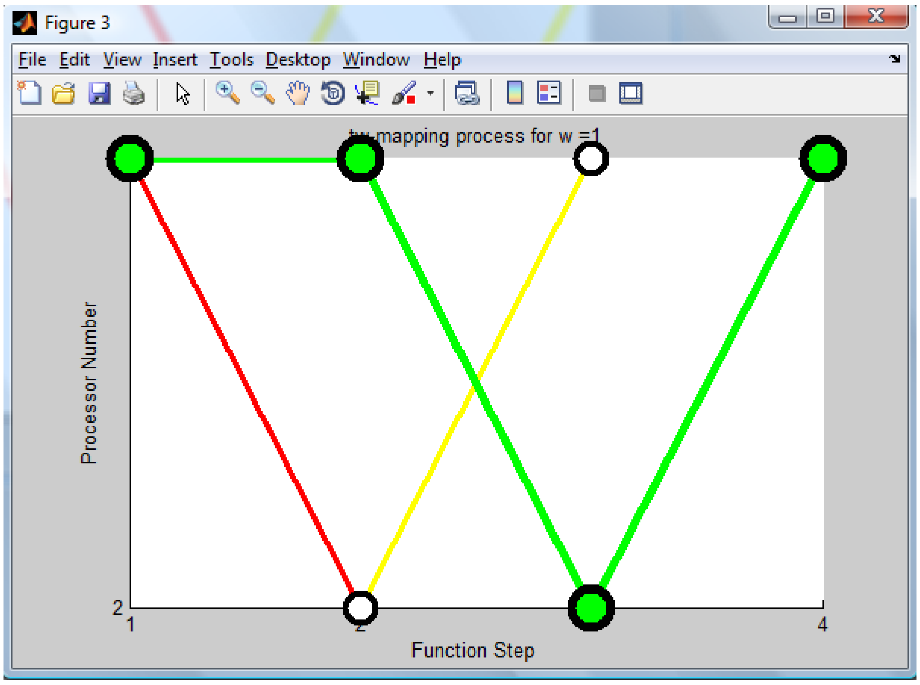


Figure 6 - t_1-mapping process illustration (parts II and III).

The minimum-cost *t*-node of this mapping example is {P_1, f_4}, the upper-right node of Figure 5 or 6. This is the origin node of the backtracking process.

You may have noticed that for the above parameter set no edge is drawn at *t*-node {P_2, f_4}. This node is actually missing in the Figures 5 and 6 (as well as *t*-node {P_2, f_1}, which is unreachable as no decision path leads to it). The missing *t*-node {P_2, f_4} here indicates that mapping SDR function f_4 to processor P_2 is not feasible at this point of the mapping process, characterized by given the previous decisions [$f_1 \rightarrow P_1$, $f_2 \rightarrow P_2$, $f_3 \rightarrow P_1$] and [$f_1 \rightarrow P_1$, $f_2 \rightarrow P_1$, $f_3 \rightarrow P_2$]. These decisions are associated with the origin *t*-

nodes {P_1, f_3} and {P_2, f_3} of the two edges that are evaluated at *t*-node {P_2, f_4}.) Either the remaining processing resources of P_2 are insufficient for executing f_4, or the remaining bandwidth resources are insufficient for getting all necessary data to or from P_2.

Redo the example for w = 3 (modifying line 33 of *aloe8a.m*) and observe the differences in the procedure and outcomes. The Command Window indicates:

```
{P2, f2} is the minimum-cost t-node at step M-w+1=2.
```

The minimum-cost *t*-node is thus {P_2, f_2}, which is the reference point for part III of the t_3-mapping process.

The window size w here coincides with the number of SDR application modules minus one (M–1 = 3). The t_3-mapping has thus explored all possible mapping combinations and found the optimal mapping solution with respect to the applied cost function.

In ALOE Session 7 you have learned that there are N^M different ways for assigning M processes to N processors. There are $N^M = 2^4 = 16$ different mappings here, 8 are associated with *t*-node {P_1, f_2} and 8 with {P_2, f_2} (Table 2). The tool captures only the decisions, that is, one out of 8 w-paths at each one of these two *t*-nodes, one in yellow ("1 0 0 1" @ *t*-node {P_1, f_2}) and the other in red ("0 1 1 0" @ *t*-node {P_2, f_2}). Part III selects the red w-path, being less costly than the yellow.

Table 2 - All mapping combinations with M=4 processes and N=2 processors.

@ *t*-node {$\boldsymbol{P_1}$, f_2}	@ *t*-node {$\boldsymbol{P_2}$, f_2}
0 **0** 0 0	0 **1** 0 0
0 **0** 0 1	0 **1** 0 1
0 **0** 1 0	0 **1** 1 0
0 **0** 1 1	0 **1** 1 1
1 **0** 0 0	1 **1** 0 0
1 **0** 0 1	1 **1** 0 1
1 **0** 1 0	1 **1** 1 0
1 **0** 1 1	1 **1** 1 1

Note that the complexity of computing all possible mappings grows exponentially with the problem size [1]. It is therefore not practical to search for the optimum mapping solution for reasonable problem sizes; M > 15 and N > 2, for instance.

5.5. Scheduling

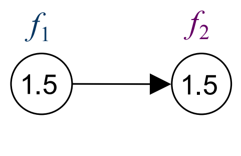
The mapping ensures that the available computing resources are allocated as needed. The scheduler then decides on the execution order of modules and schedules the processing tasks and data flows within a single time slot. It needs to ensure that the processing and propagation of data finishes within the time slot boundaries. This is discussed in [1] and [2].

Many scheduling approaches are possible. In this session we examine a simplified version of the scheduling problem, dealing with the scheduling of the processing tasks only. That is, rather than coordinating the data processing and propagation, we illustrate the simplest form of scheduling, where the application modules are executed in logical order.

The basic scheduling concept assumes a delay of one time slot for each data flow dependency [1] [2]. If process #1 (f_1), for example, sends data to process #2 (f_2) the data packet that f_1 processes in time slot x is processed by f_2 in time slot x+1. While f_2 processes the first data packet, f_1 processes the second (pipelining). That is, all processing tasks occur in parallel, process several data blocks in a pipeline. Figure 7 illustrates this for this basic example, where f_1 and f_2 are mapped to processors P_1 and P_2, respectively. The scheduling diagram of Figure 7b captures three time slots. It indicates that f_1 consumes 25 % of the processing resources of P_1, whereas f_2 consumes 50 % of the processing resources of P_2. The two SDR functions may have the same processing requirement (1.5 MOPTS due to Figure 7a), in which case the processing capacity of P_1 doubles that of P_2.

The processing resources and requirements are specified in million operations per time slot (MOPTS) [1] [2]. Processes are invoked once per time slot. The percentage of processor utilization can then be translated to processing time per time slot of known duration. Time is commonly used as the basic unit of scheduling diagrams.

If both processes of the above example were mapped to the same processor, they could process the same data packet in a single time slot, minimizing the pipelining latency. These and other more sophisticated scheduling options are beyond the scope of this session.



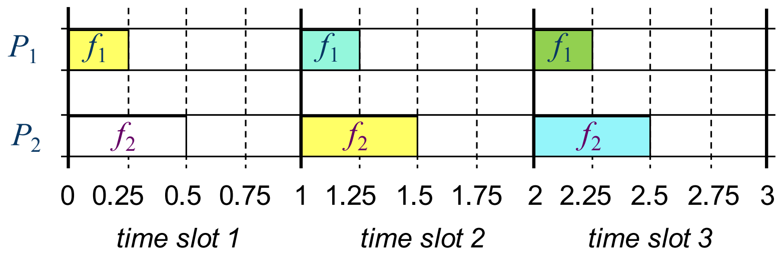


Figure 7 - Simple data flow dependency (a) and the corresponding scheduling diagram (b).

Type

```
aloe8b
```

in the Matlab Command Window. Observe the outcomes, which are captured by Figures 8 and 9 below.

Eight processes are mapped to processor #1 (P_1), four processes to processor #2 (P_2), and the remaining twelve processes to processor #3 (P_3). These processes correspond to the UMTS receiver processing modules, modeled in *umts1.m* and illustrated in Figure 10 in the appendix. (The same model is used in ALOE Session 7 [2].) The process scheduling is shown in the "*scheduling diagram*" (Figure 8 below). The x-axis is marked from 0 to 1 and represents a single time slot. (The time slot duration is irrelevant here.) You may need to zoom in to notice some of the smaller modules, those that require very few processing resources or processing time per time slot.

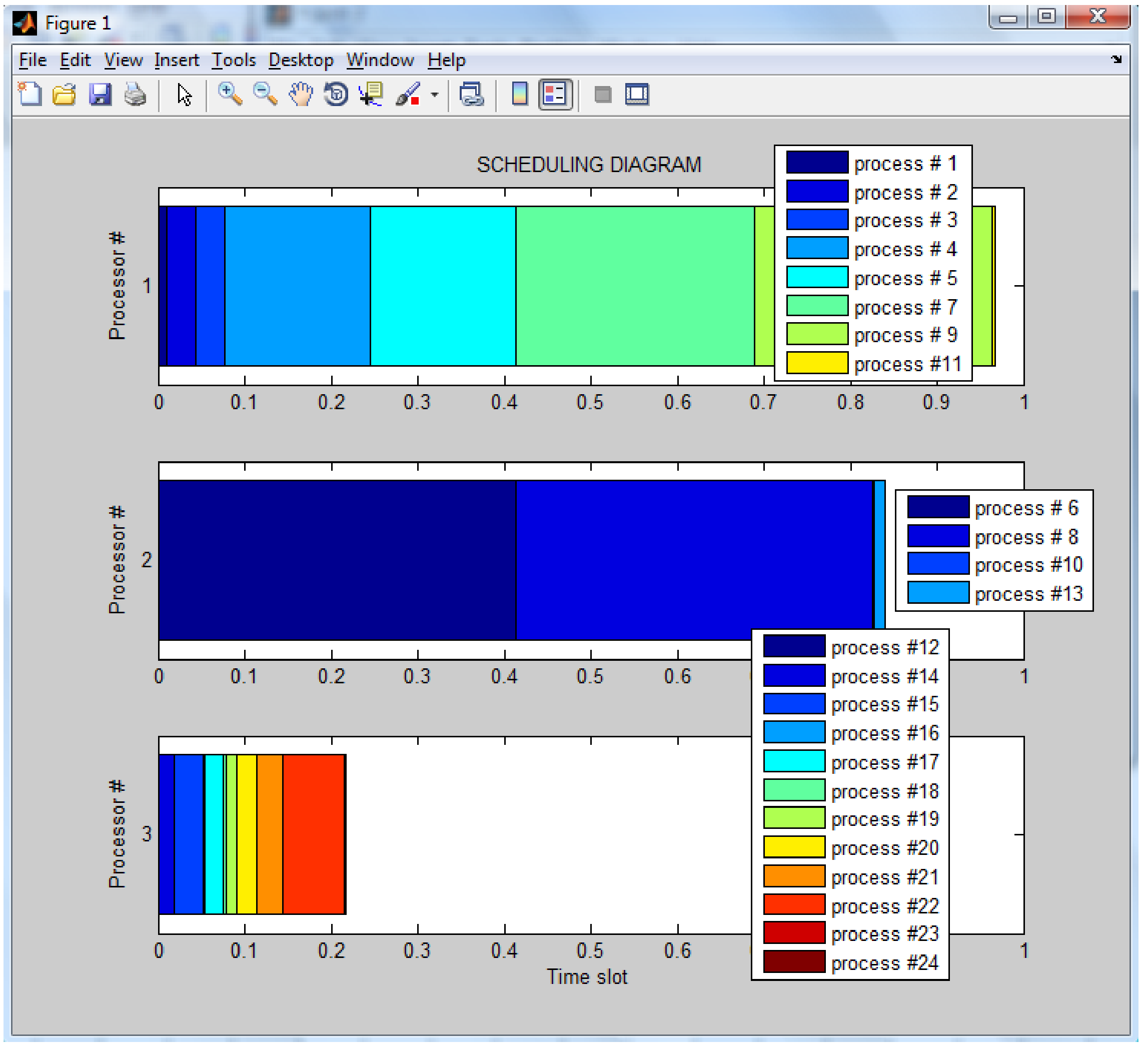


Figure 8 - Scheduling diagram for *c_load* = 0.8, *b_load* = 0.5 (alg=1, w=1, q=0.5, k=0, order=0, mhop=0).

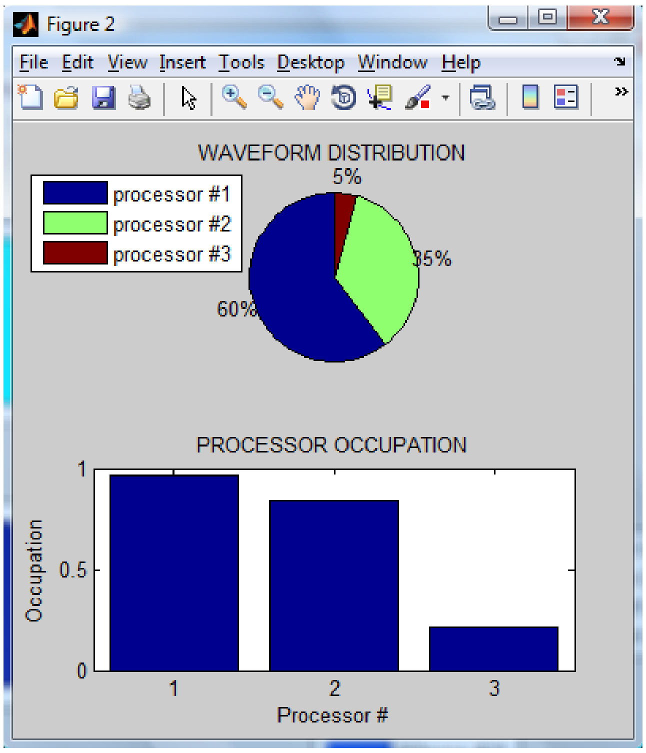


Figure 9 - Waveform distribution and processor occupation for *c_load* = 0.8, *b_load* = 0.5 (alg=1, w=1, q=0.5, k=0, order=0, mhop=0).

The "*waveform distribution*" subplot shows how the waveform's total processing requirement is distributed among the available processing resources. P_1 here executes 60 % of the waveform's total processing load, P_2 35 % and P_3 5 % (Figure 9).

The "*processor occupation*" subplot finally shows that P_1 and P_2 are highly occupied, which corresponds to the few idle processing resources/time per time slot (white regions in Figure 8).

Change *c_load* value to 0.5 (line 21 in *aloe8b.m*). Save and execute the script again. Observe the differences. Because of the low processing load, P_3 remains idle. That is, no application module is mapped to P_3; the empty scheduling window and the 0 occupation of processor #3 confirm this.

6. *Exercises*

1. Consider the following platform and application models:
   - ***C*** = (4, 4),
   - ***B*** = $\begin{pmatrix} \infty & 1.3 \\ 1.6 & \infty \end{pmatrix}$,
   - ***c*** = (2, 1.7, 0.75, 0.5)
   - ***b*** = $\begin{pmatrix} 0 & 0.75 & 0 & 1 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Compute the g_{w^-}, t_{w^-} (w = 1, 2, 3) and the *opt*-mapping using the MEX-file.

2. Does increasing the window size necessarily lead to a better (lower-cost) mapping solution? Reason your answer.

This finishes ALOE session 8. Please send your feedback to flexnets.pmt@upc.edu.

References

[1] V. Marojevic, "Computing Resource Management in Software-Defined and Cognitive Radios", Ph.D thesis, Universitat Politècnica de Catalunya, 2009. Available online: http://flexnets.upc.edu/trac/wiki/Publications

[2] V. Marojevic, Ismael Gomez, Antoni Gelonch, "ALOE Session 7: Computing Resource Management", Universitat Politècnica de Catalunya, June 2011. Available online: http://flexnets.upc.edu/trac/wiki/ResourceManagement

[3] Vuk Marojevic, Ismael Gomez, Antoni Gelonch, "The FlexCRM Project", Universitat Politècnica de Catalunya, July 2011. Available online: http://flexnets.upc.edu/trac/wiki/ResourceManagement

[4] D. E. Kirk, "An Introduction to dynamic programming," *IEEE Trans. Education*, vol. 10, iss. 4, pp. 212-219, Dec. 1967.

Appendix

Table 3 - Computing resource management tools (open-source files).

File	Description
mapper_matlab.mexw32	Mex-file that permits executing the mapping algorithms (compile in C) from Matlab. The available version runs on Windows, 32 bits.
aloe8a.m	Defines a simple mapping problem, calls the mex-file and the tw-mapping process animation routine.
aloe8b.m	Defines another mapping problem, calls the mex-file and executes the other CRM tools.
trellis.m	Function that animates the premapping decisions of part II and part III of tw-mapping process. It takes as the inputs, the highlighted edges, the window size w, and the processor index of the minimum-cost t-node at step $M–w$.
trellis2.m	Does the same as the *trellis* function, but stops after finishing with the animation of part II.
get_y.m	Supporting function of the *trellis* and *trellis2* functions
resource_models_ transformation.m	Transforms the interprocessor communication model
umts1.m	UMTS receiver model

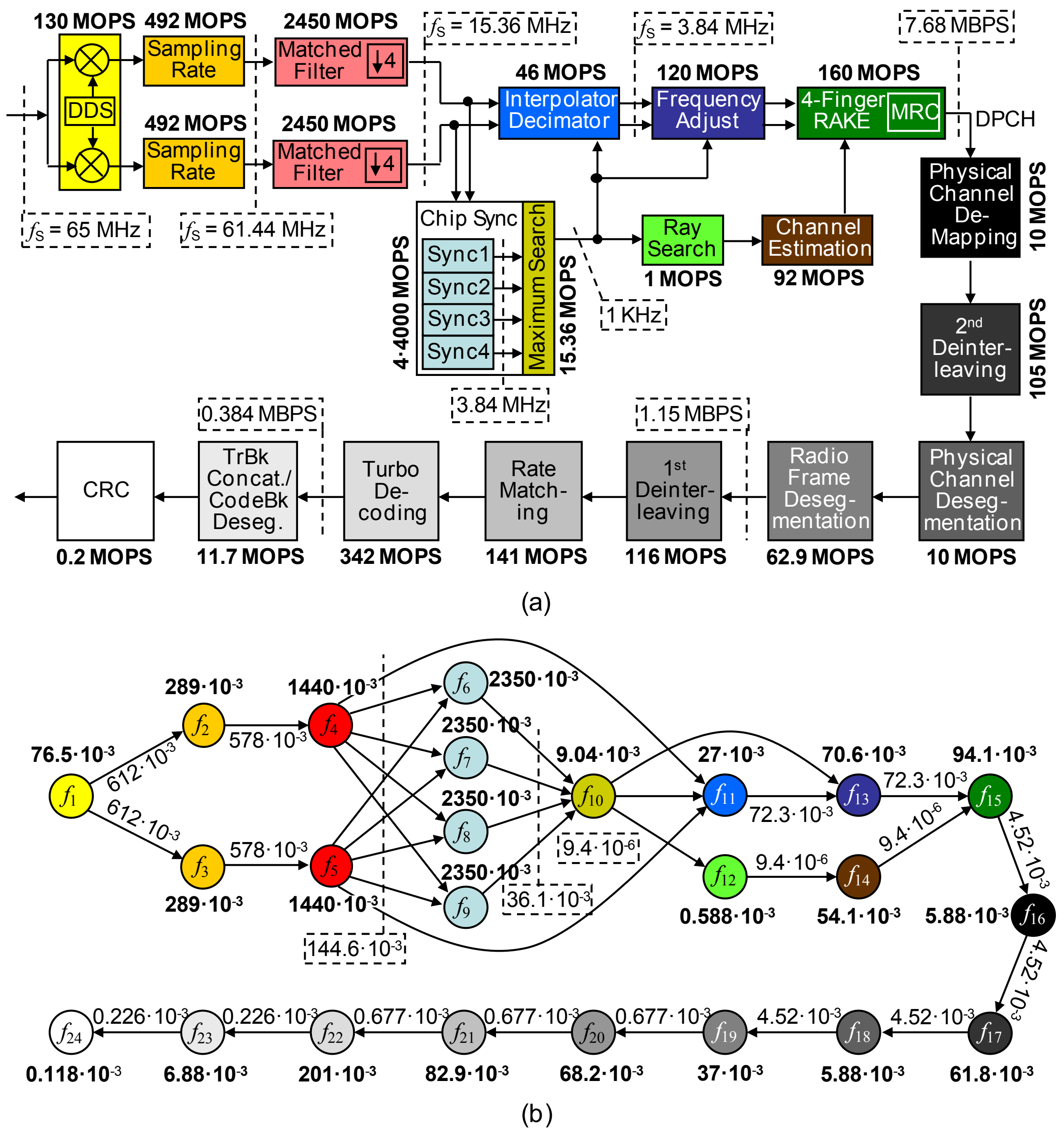


Figure 10 - UMTS receiver processing chain and modeling parameters (a). SDR application model (MOPTS and MBPTS) for a time slot duration of $0.588 \cdot 10^{-3}$ s (b).