Optimisation des ressources matérielles et logicielles d'un drone engagé dans une mission d'interception

Julien Mazuet

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Université de Bretagne Occidentale

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Électronique*

Par
Julien MAZUET

Reconfiguration des ressources matérielles et logicielles d'un système radar embarqué en mission d'interception

Thèse présentée et soutenue à l'Université de Bretagne Occidentale, le 05/10/21
Unité de recherche : UMR6285 - Lab-STICC

Rapporteurs avant soutenance :

Gilles SASSATELLI	Directeur de recherche, LIRMM
Xavier NEYT	Professeur, Royal Military Academy

Composition du Jury :

Président :	Christophe JEGO	Professeur, Enseirb-Matmeca
Examinateurs :	Gilles SASSATELLI	Directeur de recherche, LIRMM
	Xavier NEYT	Professeur, Royal Military Academy
	Jean-Marc LECAILLEC	Professeur, IMT Atlantique
Dir. de thèse :	Jean-Philippe DIGUET	Directeur de recherche, CNRS
Co-enc. de thèse :	Catherine DEZAN	Maître de conférence, UBO

Invité(s) :

Arnaud PHELIPOT	Ingénieur R&D matériel radar, Greenerwave
Alexandre SKRZYNIARZ	Ingénieur spécialiste logiciel, Thales DMS

Acknowledgement

Embarking on a PhD is a truly life-changing experience, one that would not have been possible without all the support and guidance that I received.

First and foremost, I would like to thank Gilles Sassatelli, research director at the LIRMM, and Xavier Neyt, Professor at the Royal Military Academy for agreeing to be the thesis rapporteurs for this work. Their work helped me to clean my manuscript to the state it is today.

I would also like to thank Christophe Jego and Jean-Marc Lecaillec, respectively Professor at the Enseirb-Matmeca and Professor at the IMT Atlantique, for their participation in the jury. Their questions were insightful and opened new outlooks for my work I could not have thought of alone.

I would then like to thank my PhD supervisors, Jean-Philippe Diguet and Catherine Dezan, for their unflinching guidance and support througout my three years of thesis.

A doctorate like I did is made thanks to my university, but also thanks to the company and its people who welcomed me there. I would like to thank my former Thales supervisor, Michel Narozny, for he taught me a lot about the domain of radar which was unknown to me.

I would also like to thank my second Thales supervisor, Arnaud Phelipot, who showed great interest in my work and provided me with great help during the final year of my thesis.

I will never thank enough the people who helped me increase my knowledge and experience during this PhD: Dominique Heller for sharing his experience of hardware architectures and design tools, Pierre Clouet for his wise advice on radar hardware architectures and Jean-Marc Lecaillec for sharing his knowledge of radar signals modeling and processing.

I would like to thank my colleagues and friends who accompanied me on this journey: Alexandre, Alix, Camille, Cédric and Cédric, Damien, Éloïse, Erwan, Johann, Florent, Katy, Flavien, Gabriel, Mathilde, Maxime, Paul, Thomas, Titouan, Yohann, Zoé and all of the others whom I may have forgotten (all my apologies for that).

I thank my family whose support has been very helpful throughout my entire life.

In particular, I wish to thank my parents, my brother Rémi, my sister Margot and her husband Keryan.

I thank my best mate, Benoît Rivot, who accompanied me (and whom I accompanied) during my college studies, without whom I would not have gone so far, either in Taiwan or in this thesis.

Finally, those acknowledgment wouldn't be complete without this one, the most important. I thank my other half, Clémantyne Aubry, for her endless support despite the ups and downs that punctuated the last three years. She made my life brighter through this hard but rewarding thesis, and I hope to be able to support her on her work as much as she did to me.

Table of Contents

INTRODUCTION

Since their invention in the early 1900s, RAdio Detection And Ranging (radar) became widely used, initially for defense applications, before finding numerous applications in the civilian sector. Nowadays, radars have applications in many fields, such as self-driving cars, weather forecasting and geological observation. Historically, radars were composed exclusively of analog parts which have great cost-performance ratios, but lack of flexibility and integration. Indeed, although complex functions could be implemented, they required heavy hardware to perform them. For example, obtaining Doppler frequencies, and thus velocities, required the implementation of many parallel analog filters, whereas the same result can now be obtained by Fast Fourier Transform in digital components. The development of digital technologies (e.g. microprocessors, digital-to-analog converters and programmable logic devices) led to the gradual replacement of analog components for digital signal processing, as illustrated in Figure 1. This transition allowed to use more complex algorithms as well as the creation of modes as diversified as tracking, imaging and radio-communication within the same radar system. In addition, some upstream operations such as down- and up-conversion (demodulation and modulation up to the X band) can now be performed by digital components for a better integration.

Nowadays, the integration and miniaturization of radar systems allow the implementation of many transceivers in parallel. The use of transceivers array allows the electronic steering of the antenna: the antenna array does not have to move physically to look in a direction, hence it can look in several directions simultaneously. In addition, the capabilities of analog-to-digital converters and computing systems allow the parallel computation of these channels that can be efficiently implemented in FPGA. The modern radars are electronically steered and multi-modes, such as those shown in Figures 2, 3 and 4.

However, the resulting increase of radar data and modes brings many challenges in embedded systems, where the available computing resources are highly constrained. Specifically, embedded systems have limited power consumption, computational resources and are often subject to real-time constraints. Meanwhile, the need for performances and the new challenging technologies, such as stealth aircraft, are keeping up the demand for ever more capable systems. Of course the computing technologies and architectures evolve as

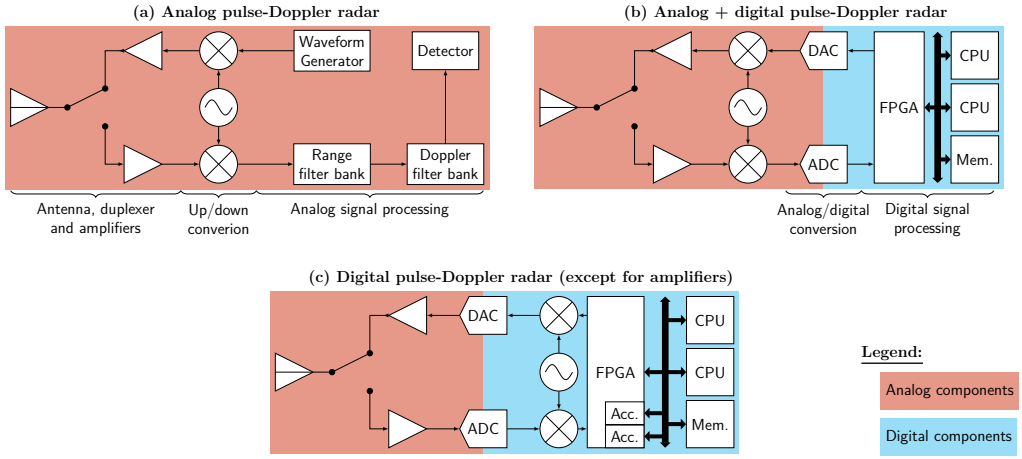


Figure 1 – Evolution of pulse-Doppler radars: from an analog sensor (a) to the common analog/digital hybrid solution (b) to a fully digital platform (c). The illustrations show a single antenna for the sake of simplicity. In practice, many antennas can be implemented, so the functions would be repeated for each antenna. An additional up/down conversion stage may be necessary depending on the operating frequency of the radar. The more functions are performed by digital components, the better the integration of the radar system.

well, but they are not up to the challenge. Indeed, Moore's law alone can not explain why the embedded computing power keep increasing since it suffers issues such as power consumption [1] and heating [2] which limit the performance increase. Further increasing computing power is possible using specialized computational units, i.e. dedicated functions implemented in hardware which are more efficient (in terms of throughput, latency and energy consumption) than general processors. They can be implemented in ASIC and FPGA to offload complex functions from the CPU, but have a footprint. Such heterogeneous architecture, where CPU are assisted by specialized accelerators is presented in Figure 1 (c). With the diversity of modes required by modern radar systems, the concurrent implementation of many functions is not scalable because of their cumulative footprint. As opposed to ASIC, the specialized functions implemented in FPGA can be reconfigured to implement new functions, giving a solution to the scaling problem. To achieve the best performance at a given time with given resources, we must be able to create systems which make the most of the resources available at runtime.

1. Pollack's rule states that the performance of processors grows is roughly linearly proportional to the square root of complexity, while power consumption increases linearly with complexity [1].

2. The dark silicon phenomenon, for example, is the fact that some parts of the microchips silicon remain unused due to thermal problems.

Figure 2 – MPAR radar: the multi-mode phased array radar of National Severe Storms Laboratory. Source: `https://www.nssl.noaa.gov/tools/radar/`

Figure 3 – An array antenna radar for detection and avoidance produced by Hensoldt. Source: `https://www.hensoldt.net/news/hensoldts-collision-warning-system-for-drones-ready-for-take-off/`

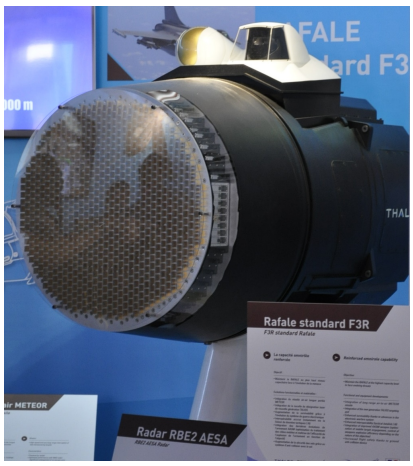


Figure 4 – RBE2 radar model: a multi-transceiver and multi-mode airborne radar produced by Thales. Source: `https://fr.wikipedia.org/wiki/Thales_RBE2#/`

Hopefully, radar does not need to use every functions and modes simultaneously during the mission. In addition, computing resources can often be reconfigured with varying degrees of difficulty. A solution to keep up performance in this constrained situation is to adapt the computing resources usage to the context of the mission. A common optimization technique is based on the reconfiguration of the radar mode by changing large code portions and/or FPGA configuration. This is done by switching from one configuration to another, for example from aircraft to weather tracking. These reconfigurations concern a large part of the system and are bulky, which is acceptable for systems running in a stable environment, like ground radars, but is prohibitive for highly dynamic systems like UAV and aircrafts. However, FPGA also allow partial reconfiguration, i.e. reconfiguring a small part of the system while the rest of the system continues to operate. This solution reduces the reconfiguration time to an acceptable level, but remains underexplored in radar processing literature.

In this PhD work, we explore the reconfiguration capabilities of embedded radar tracking systems evolving in an uncertain environment. The constraints of embedded and real-time systems require solutions that overcome the problem of reconfiguration time. Based

on our studies, we come up with the idea that both architecture and algorithm adaptation must be considered to benefit from hardware reconfiguration under resource constraints. So we propose contributions in both domains and a design methodology approach to combine the different solutions in a heterogeneous industrial context. We propose solutions based on the dynamic partial reconfiguration of FPGA and algorithm adaptations. These solutions are inspired by problems identified at Thales in the development of embedded radar systems. Developing such reconfigurable systems is complex since it requires hardware, software and application knowledge, which led us to search methodological elements to design such systems. In this thesis we demonstrate the optimizations that can enhance embedded radar systems with help of dynamic reconfiguration and then we define a methodology to efficiently design reconfigurable radar systems across several technical experts.

Thesis contributions

1. Description and test of two original radar case studies of optimization, inspired from limitation of radar computing systems observed at Thales, based on hardware dynamic reconfiguration:
    - Tracking reconfigurable architecture: Two reconfigurable accelerators are implemented. These accelerators can be used to run different tracking models and an algorithmic indicator can detect if the models are adapted to the tracked objects. The unadapted models are reconfigured at runtime to maximize the tracking performances.
    - Doppler processing reconfigurable architecture: Two algorithms can be used for the extraction of Doppler frequencies in a radar signal. These algorithms have different properties and this architecture adapts the tradeoff between processing latency and quality by reconfiguring the current algorithm depending on the mission context. The reconfiguration decision is based on an algorithmic indicator. This architecture is validated by a hardware-in-the-loop simulation.
2. Revisit, modeling and test of an algorithm (the Space-Time Adaptive Processing) adaptation, based on machine learning, to reduce execution time while keeping up the performances. This algorithm was first tested in software, but is now compliant and ready to use within a reconfigurable architecture high performance implementation.

3. Definition of a methodology for the design of reconfigurable systems based on SoC. A set of design patterns and rules are defined to simplify the design of reconfigurable systems. A case study of a development based on the two hardware dynamic reconfiguration case studies illustrates the methodology.

Thesis organization

This thesis is composed of six additional chapters [3] ordered as follows:

Chapter 1: Research background: This chapter details important concepts used in this thesis, so that the reader can approach the rest with a clear understanding of the notions and stakes of radar systems reconfiguration. The considered radar, processing and computing systems are exposed and the reconfiguration possibilities of radar system are explored. This chapter concludes on the advantage of reconfiguration in the radar domain examples that introduce the rest of this thesis.

Chapter 2: State of the art: This second chapter is divided into four sections. The first section describes use-cases of dynamic partial reconfiguration and presents examples of radar systems DPR-based optimizations. The second section details the three algorithms on which we based the reconfigurable systems presented in chapters 3 and 4. The third section is about reconfigurable systems development methods and to the multi-agent paradigm. The last section positions the thesis work in the literature.

Chapter 3: HW Reconfiguration based on Algorithm choice: This chapter is composed of two parts for the two case studies of FPGA-based reconfigurable radar applications. The two sections detail the concepts and architectures created for these case studies (contribution n°1).

Chapter 4: Algorithm adaptation to benefit from reconfiguration: This chapter describes a machine learning system which can predict the most adapted dimensions reduction of a Space-Time Adaptive Processing at runtime, to maximize quality of results while reducing computing complexity (contribution n°2).

Chapter 5: Methodology for reconfigurable systems design: This chapter proposes a methodology for the design of reconfigurable radar systems. Based on

3. Several sentences of Chapter 2 are inspired or imported from published ([2], [3]) and submitted articles. Chapter 3 is composed of papers published in the context of this thesis with references [2], [3] for the sections 3.1 and 3.2, respectively. I am the primary author of the articles used in this thesis.

observations of the case studies presented in Chapter 3, we draw methodological elements which should ease the design of a reconfigurable radar system by a team of heterogeneous developers (contribution n°3).

Conclusion and future work: This last chapter summarizes the work done in this thesis and outlines perspectives for future work.

Chapter 1

Research background

Contents

In this chapter, we focus on the signals and tools used in the context of this PhD research. More specifically, Section 1.1 details the radar waveforms and antennas considered in the studies as well as the conventions used in the radar equations. Section 1.1 also describes the most common algorithms used in a radar tracking system. Section 1.2 describes the reconfiguration capabilities of computing resources commonly available in embedded radar systems (e.g. in UAV systems). Then, it presents the tools and methods associated with these reconfiguration means. Finally, Section 1.3 concludes on the reconfigurability of radar processing systems.

1.1 Radar

1.1.1 Signal description

There are two main categories of radar signals: pulsed and continuous wave. Both of these signals allow to have high range and Doppler resolutions. In this thesis, we use pulsed signals as they are more resilient to interferences (e.g. clutter and jammer) compared to frequency modulated continuous waves (FMCW).

The simplest pulsed waveform is a monochromatic signal, (i.e., it can be defined by a single frequency). With this waveform, we can achieve a range resolution $\Delta r = \frac{c}{2}\tau$, where c is the speed of light in the air and τ is the duration of the pulse. However, to obtain a long detection range, we need to maximize the pulse energy. To increase the pulse energy, we need to send longer pulses, which decreases the range resolution obtained with a monochromatic wave. To alleviates this problem , the waveform used in this thesis is the linear-frequency chirp pulse. This signal frequency increases linearly with time to cover a frequency bandwidth B. The matched filter of the linear chirp (called chirp/range compression) compresses the chirp bandwidth and outputs a cardinal sinus with a peak of power at the location of the central time of the echoed chirp. The output of this matched filter has an expected gain of $\tau \cdot B$ on the signal-to-noise ratio (SNR). The attainable range resolution with this waveform is $\Delta r = \frac{c}{2B}$. The range resolution is not dependent on the pulse duration anymore so we can have both a good range resolution and a high energy by spreading the power over the full chirp bandwidth. In the case of pulse-Doppler radars, the signal is the chirp and is repeated periodically (and coherently) after a period $T_r = 1/f_r$ (f_r is called the Pulse Repetition Frequency or PRF) and is active during a portion of this time τ, with $\tau < T_r$. This pulsed signal, of amplitude A, can be defined by

the function:

$$x(t) = \begin{cases} Ae^{2\pi j(f_0 + \frac{B}{\tau}t')t'} & \text{if } 0 \leqslant t' < \tau \\ 0 & \text{else} \end{cases}, \text{ where } t' = t \mod T_r \tag{1.1}$$

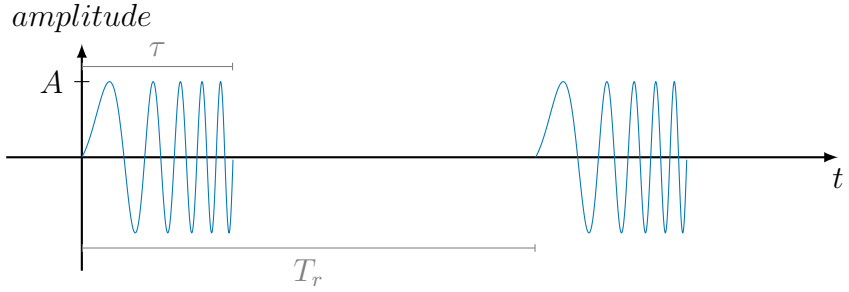


Figure 1.1 – Example of a pulsed chirp signal

This signal is emitted by the radar, reflected by a target and propagates back to the radar with a delay δt, a frequency shift f_{Dopp} (Doppler) and noise plus interferences n. It can be synthesized as:

$$y(t) = A'x(t + \delta t)e^{j2\pi f_{Dopp}t} + n(t) \tag{1.2}$$

where A' is the modification of the signal amplitude at reception, which aggregates several physical parameters such as the radar cross section of the target and radar-to-target distance.

This signal is described for a single channel or antenna and does not contain information about the direction of arrival of the wave. However, this spatial information is essential to locate the targets and improve signal processing gains. This can be achieved with multi-element antennas such as the AESA.

1.1.2 Active Electronically Scanned Array (AESA)

In modern radar applications, the antenna has usually multiple transceivers. This enables the electronic steering of the antenna which is quicker and more reliable than a mechanical steering, while allowing to scan many directions at the same time. Moreover, these transceivers arrays allows the use of the antenna for different purposes at the same time, such as communication and tracking. The processing can take advantage of the spatial information brought by the signal arriving at different times on the antenna sensors

to perform spatial filtering. Figure 1.2 depicts a linear phased array receiving a signal from a θ elevation. We see that the wavefront is not aligned with the antenna and that the information will arrive on the different sensors with a phase shift of $\frac{2\pi}{\lambda} d \cos(\theta)$ rad often referred to as spatial frequency.

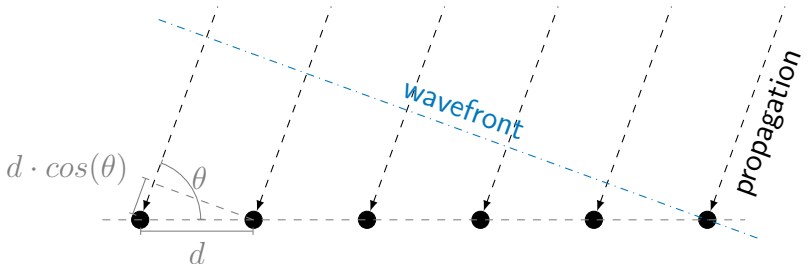
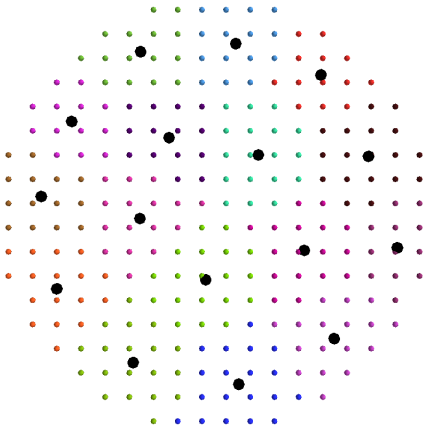
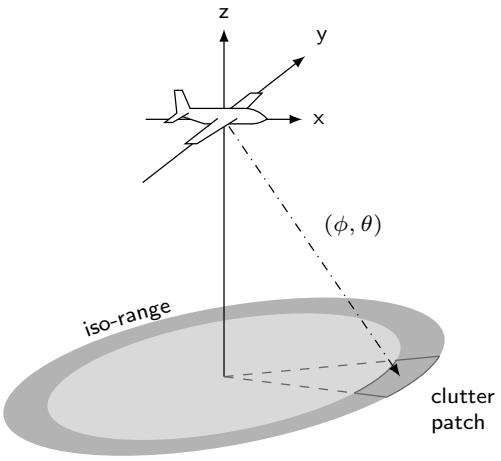


Figure 1.2 – Radar wave propagation viewed from a 1D AESA

AESA comes in two forms: the planar active electronically scanned array (AESA), and the conformal antenna. Conformal antenna sensors cover a curved surface, for instance the skin of an aircraft. Their study is complex since the multidimensional coupling between the sensors must be compensated by electronic phase shifts. To avoid unnecessary problems, we use a planar AESA such as the one presented on Fig. 1.3a. The colored dots represent hundreds of antenna elements, that we cannot process in parallel due to processing resources limitations. Hence, we group them into clusters represented by the different colors. These clusters are associated with a phase center, represented by the black dots.



(a) Example of theoretical 2D AESA



(b) Geometry of airborne radar (adapted from [4])

AESA can be steered electronically and/or numerically depending on the underlying implementation of the system. The theoretical array factor (gain provided by the electronic steering) of such an antenna can be easily computed, with a scalar product of the steering vector $G_{\theta,\phi} = \mathbf{s}^H_{\theta,\phi}\mathbf{s}_{\theta_a,\phi_a}$ with θ, ϕ and a denoting the elevation, the bearing and the aiming direction of the antenna, respectively. Figure 1.3b shows the geometry involved in the airborne radar system and the creation of steering vectors. The steering vector $\mathbf{s}_{\theta,\phi}$ contains the difference of phase between the different sensors, for a signal arriving from a direction defined by θ and ϕ. It can be written as:

$$\mathbf{s}^T_{\theta,\phi} = [e^{j\varphi_0(\theta,\phi)}, e^{j\varphi_1(\theta,\phi)}, ..., e^{j\varphi_N(\theta,\phi)}] \tag{1.3}$$

where $\varphi_n(\theta,\phi)$ is the phase shift (relative to a reference point) for element n. In the 1D antenna of Fig. 1.2, Eq.(1.3) expands to:

$$\mathbf{s}^T_{\theta,\phi} = [e^{j\frac{2\pi}{\lambda}d\cos(\theta)\sin(\phi)}, e^{j2\frac{2\pi}{\lambda}d\cos(\theta)\sin(\phi)}, ..., e^{jN\frac{2\pi}{\lambda}d\cos(\theta)\sin(\phi)}] \tag{1.4}$$

Figure 1.4 shows the theoretical gain of the antenna presented in Fig. 1.3a.

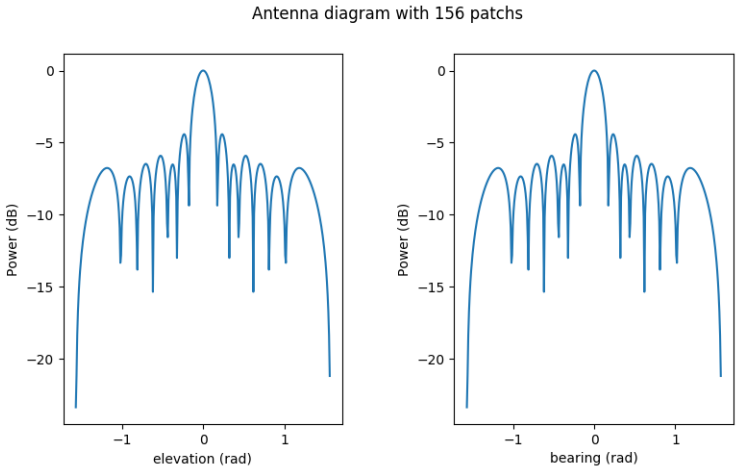


Figure 1.4 – Antenna theoretical array factor

The signal $y(t)$ received from a specific direction on an AESA can be expressed by:

$$\mathbf{y}(t) = y(t).\mathbf{s}_{\theta,\phi} \tag{1.5}$$

In practice, since the antenna is imperfect, the exact steering vectors are unknown. Approached steering vectors can be measured during a factory calibration and stored

in the computing system. They can be corrected online, to compensate imperfections or stress degradation of electronic components, by hardware or algorithmic auto-calibrations.

1.1.3 Tracking radar processing

This thesis focuses on embedded tracking radar processing. This section presents the most common signal processing algorithms used for tracking with a pulse-Doppler AESA radar. The considered input signal is digital and represented with in-phase and quadrature components (I/Q samples).

Frequency shift

At emission, the carrier frequency of the radar signal can range from a few MHz to a hundred GHz. At reception, down-conversion can be performed with analog components. However, high frequency up/down-conversion are often performed in several steps which required many components. This is an issue for UAV-borne which requires the electronic subsystem to be small and power efficient. To overcome this problem, we perform only a part of the (de)modulation with analog components and to perform the remaining frequency shift in the digital system. Between the analog and the digital modulation, the signal is modulated by what is called an intermediate frequency. The Analog to Digital Converters (ADC) must be able to perform acquisition at a frequency equal to at least two times the maximum frequency of the signal [1] modulated by the intermediate frequency. This high sampling rate results in a large amount of data to process. However, since the relevant information lies in the chirp frequency bandwidth $[-\frac{B}{2}, \frac{B}{2}]$, the computational load can be reduced. First, we perform a frequency shift to center the bandwidth around zero (which is equivalent to a demodulation) with:

$$\mathbf{y}_{bb}(t) = \mathbf{y}(t) \cdot e^{-j2\pi f_c t} \tag{1.6}$$

where $\mathbf{y}_{bb}$ is the baseband signal vector, and $f_c = f_0 + \frac{B}{2}$ is the central chirp frequency (i.e. the intermediate frequency). We can now reduce the sampling rate, but this can result in aliasing. To avoid signal distortion, we perform low-pass filtering before down-sampling the signal. The cutoff frequency of the low-pass filter must be greater than $\frac{B}{2}$. After this

1. Some techniques enable the use of a lower acquisition rate through frequency planning, but this is out of the scope of this PhD work. See `https://training.ti.com/frequency-and-sample-rate-planning-understanding-sampling-nyquist-zones-harmonics-and-spurious`.

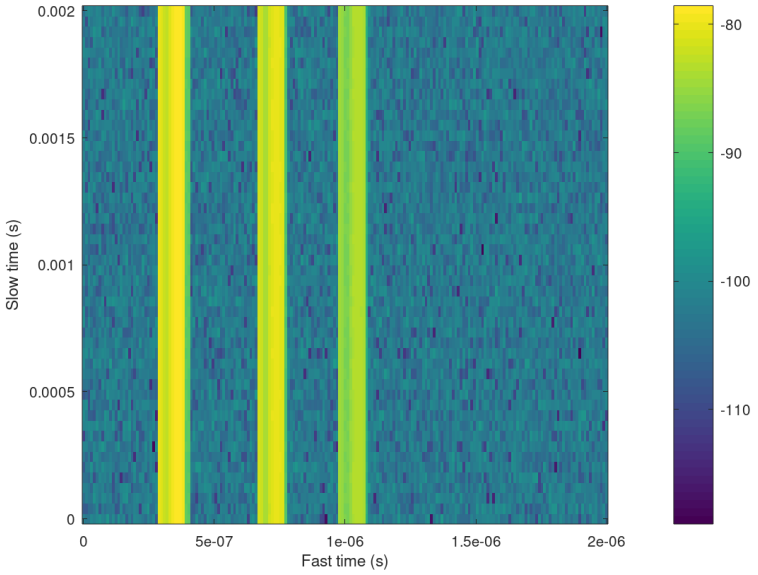


Figure 1.5 – Baseband radar signal 2D visualization (dB)

filtering step, we select a new sampling frequency to decimate the digital signal. Figure 1.5 shows such a signal which now contains the minimum number of samples needed to keep the useful information. This signal is presented as a matrix (a common representation of Doppler radar signals) where the data arrives from bottom left to top right, line after line. The fast time step is equal to the sampling period and the slow time step is equal to the pulse repetition period T_r. We can see that three targets backscatters (in yellow in Fig. 1.5) are present in the signal. However, noise and interference still coexist with this information. These perturbations can be thermal noise and other radio-waves emissions sources .

As we stated, radar data is often represented and treated as a matrix. This matrix exists for each processing channel (i.e. each antenna array or subarray elements). Therefore, the total data to process can be represented as a data cube, shown in Fig. 1.6.

Range compression

Since we know the sent signal, we can improve the SNR with a filter h tuned to our specific signal known as 'matched filter'. it consists in a convolution between the received signal $y(t)$ and the emitted (or reference) signal. Since we performed frequency shift, low-pass filtering and sub-sampling on the received signal, we must apply the same

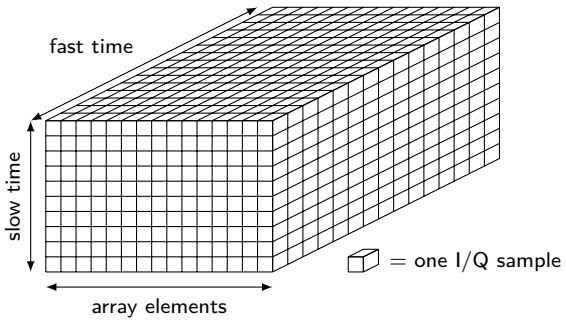


Figure 1.6 – Radar data cube

transformations to the reference signal. The filter output is:

$$\mathbf{y}_{matched}(t) = \begin{bmatrix} (y_{bb,0} * h)(t) \\ (y_{bb,1} * h)(t) \\ \dots \\ (y_{bb,N} * h)(t) \end{bmatrix} \tag{1.7}$$

where $\mathbf{y}_{matched}$ is the filtered signal vector and $h = e^{2\pi j \frac{B}{\tau} t}$ is the conjugated time-reversed reference signal (which is the baseband version of the chirp signal). Due to the convolution computing complexity, this filter is often implemented as an element-wise multiplication in the frequency domain, resulting in:

$$\mathbf{y}_{matched}(t) = \begin{bmatrix} \mathcal{F}^{-1}(Y_{bb,0} \cdot H)(t) \\ \mathcal{F}^{-1}(Y_{bb,1} \cdot H)(t) \\ \dots \\ \mathcal{F}^{-1}(Y_{bb,N} \cdot H)(t) \end{bmatrix} \tag{1.8}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. $Y_{bb,n}$ and H are the Fourier transforms of the baseband signal and the reference signal, respectively. This filter output is a signal with peaks where signal echoes are present. The time difference between the emission and the peaks location (Δt) shows how long it takes for the EM waves to go back and forth between our transceiver and the scanned objects. Since EM waves travel at light speed, this time is proportional to the distance $d = \Delta t . \frac{c}{2}$. Figure 1.7 shows the result of range compression on the signal presented in Fig. 1.5. We can now observe the distance of reflecting objects (from the yellow vertical lines in Fig. 1.7). In addition, radar also allows to extract the speed of the objects which interact with the EM waves, thanks to the Doppler effect.

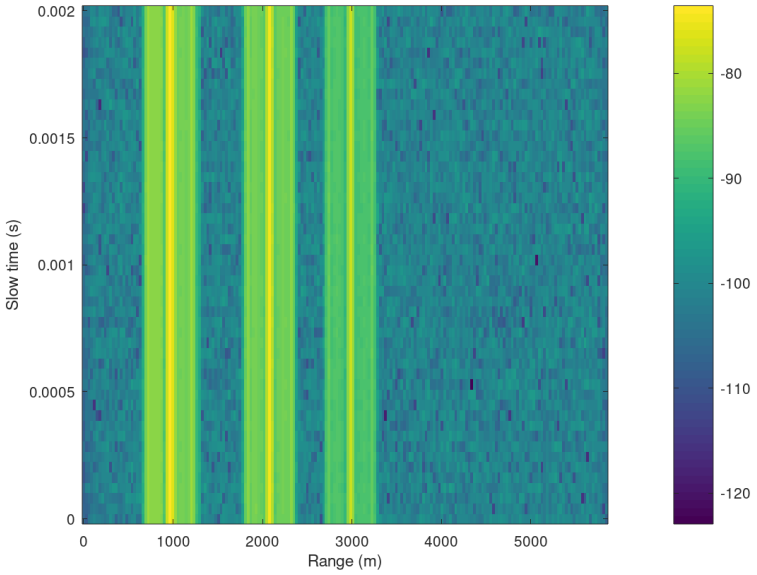


Figure 1.7 – Range-compressed baseband radar signal (dB)

Doppler processing

The Doppler effects is a signal frequency shift induced by the motion of objects relative to the wave source. To observe this frequency shift, we perform a filtering of the signal frequencies, usually performed by Fourier transforms. In the pulse-Doppler radars, it consists of sub-sampling the signal at frequency f_r and performing a DFT over this sub-signal. This process is repeated for each sample of the first pulse interval. This transforms the signal containing only information about the objects distance into range-speed maps of the scene (one per channel). Figure 1.8 shows the results of performing DFT on the signal of Fig.1.7.

The hardware implementation of this processing is highly challenging since it is not performed in the data arrival order. Indeed, we must perform a corner-turn (a data reordering operation) on a large data amount with one or several memories with limited bandwidths. This issue is further explained in Section 3.2.1.

AESA filtering algorithms

After the classic preprocessing steps, we exploit the capabilities offered by the AESA to filter out noises and interferences remaining in the signal. This kind of antenna allows further filtering operations, involving spacial filtering. The simplest spacial filter is the

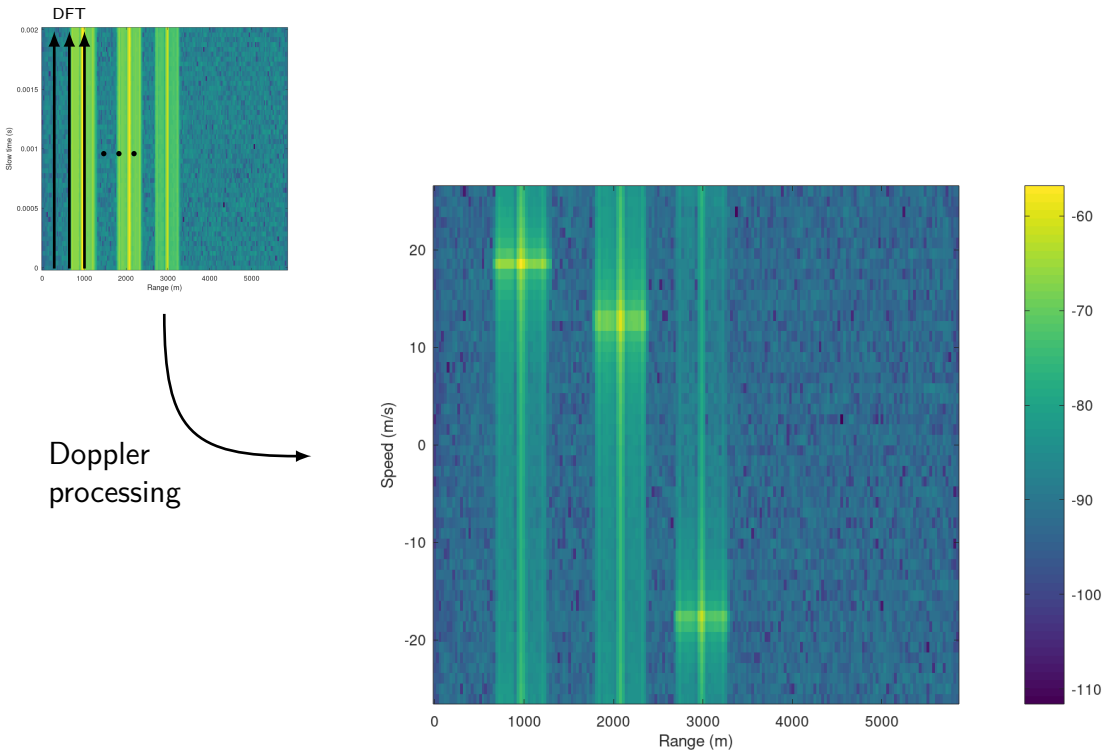


Figure 1.8 – Range-Doppler map (dB)

beamforming. It consists in multiplying the received signals vector with the transpose conjugate of the steering vector to maximize the signal coming from a direction. This is actually a matched filter, based on the phase rotation due to the delays at which the signal arrive on the different AESA sensors. The signal filtered for a couple elevation-bearing (θ, ϕ) is given by:

$$y_{\theta,\phi}(t) = \mathbf{s}_{\theta,\phi}^{H} \mathbf{y}_{matched}(t) \tag{1.9}$$

where $y_{\theta,\phi}$ is the received signal filtered to observe a specific direction. This filter theoretical gain is given by the antenna theoretical diagram (Figure 1.4). A radar antennas property is: the larger the antenna, the sharper the beam; and so is the beamforming filter response.

More complex spatial filtering can be performed at a higher computational cost. For example, the adaptive beamforming algorithm uses the background signal covariance to discriminate and filter out jamming interference. Similarly, the Space-Time Adaptive Processing (STAP) uses the space-Doppler dimension to filter out jammer and clutter (this

processing is further detailed in Section 4.1).

Detection algorithm

Once the AESA spatial filtering is performed on the input data, the signal is transformed into a range-speed map which can be seen as an image. We now find targets by searching peaks in this image. To find these peaks we define a thresholds to compare with the pixel under test. The signal SNR is *apriori* unknown, so a fixed threshold would be inefficient. Adaptive thresholds based on the noise background are created with Constant False Alarm Rate (CFAR) algorithms. Two major CFAR algorithms are the CA-CFAR and the OS-CFAR[5]. The former one computes the mean value of the pixels around the pixel under tests to set the threshold (to be compared to the cell under test) and is the simplest one. The latter sorts the pixels around the pixel under test and picks one of them (usually the $\frac{3N}{4}$th value, N being the index of the largest pixel value) to set the threshold and is more robust than CA-CFAR, especially in presence of multiple targets and/or clutter.

Both algorithms use the cells around the cell under test to compute a threshold. They can be implemented either in one dimension (flatten version of the signal), or in two dimensions (on the range-speed map). The two dimensions CFAR operations is equivalent to image processing: we use a kernel, which can be a binary cross or square, or a weighted kernel which adapts to the known target signal shape in the range-Doppler domain. Figure 1.9 presents two binary CFAR kernels examples (at the left side) as well as the detection result (with either of the kernels) on the signal presented in Fig. 1.8 (at the right side) where valid detection pixels are colored in red.

If no target is detected, the system can perform additional steps of AESA filtering and detection until a target is detected, before running the next algorithms.

Direction of arrival

When targets are detected, the system knows their range, speed and approximative direction of arrival. However, accurate angular tracking is often required. This is achieved by a direction finder algorithm which improve accuracy by orders of magnitudes. For AESA, it is possible to achieve high angular resolution with several algorithms such as MUSIC[6], ESPRIT[7] and their variants. These methods use the eigendecomposition of the measurement covariance and the orthogonality between the noise and the signal subspaces. Based on the eigenvalues, whose values are equal to the noise variance for the

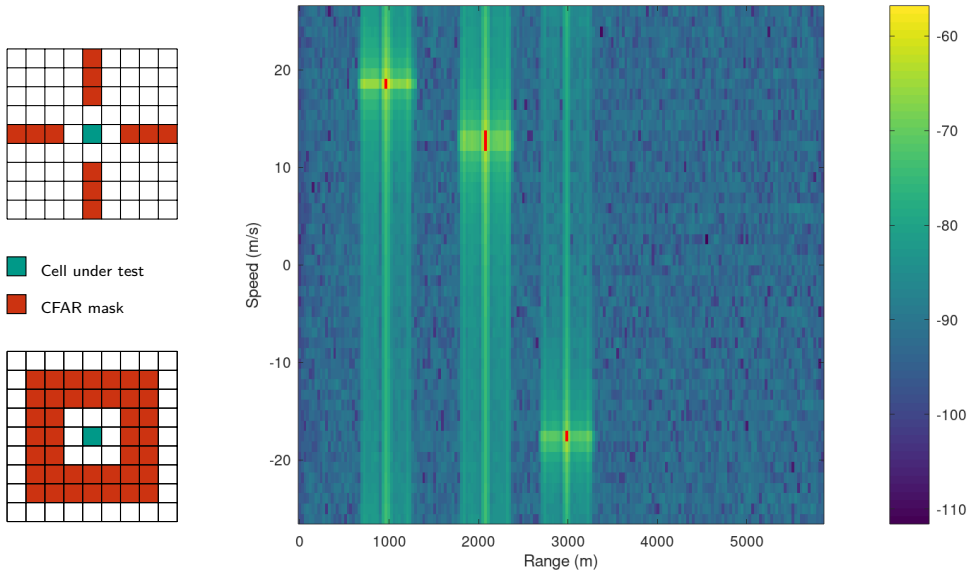


Figure 1.9 – Examples of CFAR detection mask (left) and detection results (right)

noise subspaces and greater for the signal subspaces, the signal is decomposed into signal and noise subspaces.

The MUSIC algorithm finds the directions of arrival by scanning different directions until it finds the directions where the noise subspace contains the less power, by searching the maximums of the MUSIC pseudo-specter $P(\theta, \phi)$ (Eq. 1.10), where $\mathbf{E}_n$ is the matrix of eigenvectors spanning the noise subspace.

$$P(\theta, \phi) = \frac{1}{\mathbf{s}_{\theta,\phi}^H \mathbf{E}_n \mathbf{E}_n^H \mathbf{s}_{\theta,\phi}} \tag{1.10}$$

The ESPRIT algorithm uses two separated sensors arrays (Z_x and Z_y), the second being a displaced doublet of the former one. The first step is to extract the matrix of eigenvectors spanning the signal subspace for these two arrays ($\mathbf{E}_x$ and $\mathbf{E}_y$). The second step is to solve an equation of the angular phase with the signal subspaces. Since the two arrays are related by a displacement, we have a relation between $\mathbf{E}_x$ and $\mathbf{E}_y$ such that:

$$\mathbf{\Psi} = \mathbf{E}_y \mathbf{E}_x^{-1} \tag{1.11}$$

where $\mathbf{\Psi}$ eigenvalues contain the angle of arrival of the targets (see [7] for the demonstra-

tion).

These methods rely on the antenna geometry, but as we said, the antenna is imperfect and its properties can evolve during the mission. Hence, we do not know the exact positions and phase shifts of the antenna elements which greatly condition the performance of direction finding algorithms. In [8], an algorithm based on MUSIC is presented to improve performance in the case of partly calibrated radar antennas.

Tracking and association

The targets tracking consist in filtering detected plots (that contains only informations on targets at the time of the measurement) with the help of previous measurements. One of the most common algorithms used for tracking is the Kalman Filter (KF). This filter is based on a modeling of the trajectory of an object relatively to the radar system. The implemented models are often non-linear, and extended and unscented KF (which are non-linear extensions of the otherwise linear KF algorithm) perform often better. Extended KF are preferred in embedded systems since they are less computing intensive than unscented Kalman filters. There are many ways to model a target trajectory that will perform differently depending on the observed target. The Kalman filter performance is highly dependent on the quality of the model and on the estimation of parameters such as the model and measurement noises covariance. The measurement noise can be easily estimated as we usually know the accuracy of measurement. However, the model noise can be complex to obtain since it depends on model uncertainties, but can be tuned using the innovation likelihood of the filter which gives an estimation of the model-observation consistency[9].

Some detection plots may be spurious and can be filtered out by a Probabilistic Data Association Filter (PDAF)[10]. This algorithm is an add-on to the KF, and consists in observing the innovation likelihood of the KF for each plots. If the value of the innovation likelihood drops under a defined threshold, it is considered as spurious, otherwise the detected plot is treated as a valid detection. The PDAF is also used to decide whether to create a trajectory for a series of plots.

Once the detection plots are filtered and associated to trajectories, it is interesting to classify and identify the targets.

Automatic Target Recognition (ATR)

The automatic target recognition is a process which takes inputs of different natures (speed, acceleration, ESM signature, SAR image) and returns the class of detected objects (e.g. drone, civil aircraft or military aircraft model), possibly associated with a confidence level. The inputs of ATR come from the aforementioned processing and from ancillary processing channels (e.g. the extraction of the micro-Doppler signature of the target) which can help the classifier to identify the target. The first problem is to decide which inputs, or features, must be used to classify the objects. This depends on both the available data and the application, but has to be sufficient to discriminate objects of different class. For an early-warning radar, this can be the ESM signature, speed and acceleration of the objects.

Apriori knowledge can be used to define the classification mechanisms. For example, defense companies may have access to the characteristics of many aircrafts. In some cases, we may find the range of possible classes of an object from its maneuvers. However, since we have noise and uncertainties, we cannot fix strict rules to classify objects. This rigidity can be mitigated using fuzzy logic [11].

Moderns classification methods of radar systems are intelligent and/or knowledge-based, such as HMM [12] or neural networks [13]. Several reviews of state of the art ATR systems can be found in the literature [14], [15].

The aforementioned algorithms must run in real-time in embedded radar systems equipped with various computing resources. These resources are limited, so they must be allocated to portions of the computations. In the next section, we will show that this allocation can be either static or adaptive and that adaptability can greatly benefit embedded radars performance.

1.2 Hardware/Software self-adaptive systems

In air and UAV-borne radars, the processing techniques and modes diversity as well as the evolving environment show the need for self-adaptive systems. This adaptation can be performed by means of the dynamic reconfiguration of HW and SW resources. In the following section, we present the principle of HW/SW resources dynamic reconfiguration. Then, we describe how a reconfigurable system can be characterized based on previous works on adaptive systems, with a focus on the radar reconfiguration problematic.

1.2.1 Principle

Computing systems involve various resources. In this study we focus on Systems on a Chip (SoC) composed of a CPU, a FPGA (Field Programmable Gate Arrays), memories and other resources such as communication buses. The two types of resources that we consider in our study are the computing time of CPU and the hardware resources of FPGA. Other types of resources can be considered. The dynamic reconfiguration of computing resources aims to re-attribute the available resources to the current systems needs, as opposed to a static system where the resources are attributed at design time, regardless of the execution context. A static system cannot fully benefit from the available resources since they are attributed to tasks that may or may not be useful at different time of the mission. In HW, this means that some FPGA resources will be unused during periods of the mission. Since the resources are limited, this also means that the designer must choose a tradeoff between algorithms performances (in terms of processing quality, speed or both) and resources usage. In a static system, this tradeoff will of course have higher constraints on the resources usage, and thus the performances. The reconfiguration allows to maximize the use of the resources to improve performances.

The reconfiguration of SW and HW resources are based on different mechanisms. Since SW is mostly based on the execution of sequential instructions, configuration is performed by changing this sequence. This task is fast and straightforward, so most of the challenge lies in the configuration control. However, HW reconfiguration implies change of values in many configuration memory registers, resulting in a long process. Moreover, HW architectures are often pipelined and must wait for data in pipelines to be processed before moving to a new configuration, further slowing the process. To fasten and simplify reconfiguration, we can reconfigure a subset (or partition) of the FPGA configuration. This process is called Dynamic Partial Reconfiguration (DPR).

The different reconfigurations of a system can be categorized, depending on the mechanisms used. Figure 1.10 defines three levels of reconfiguration:

1. Application reconfiguration.
2. Algorithm reconfiguration.
3. Parameter configuration.

These reconfiguration can be performed through different techniques. We categorize them as:

1. Full reconfiguration.

2. Partial reconfiguration
3. Register configuration.

The three reconfiguration levels cannot be directly mapped to one of the three configuration means. Indeed, any mean can be used for any kind of reconfiguration. The relevance of the configuration technique is application specific. According to [16], we specify reconfiguration based on a tree represented in Figure 1.11. We detail some of the following main concepts presented in Figure 1.11:

- Reason.
- Level.
- Time.
- Technique.
- Adaptation Control.

in the following sections, with a focus on radar systems.

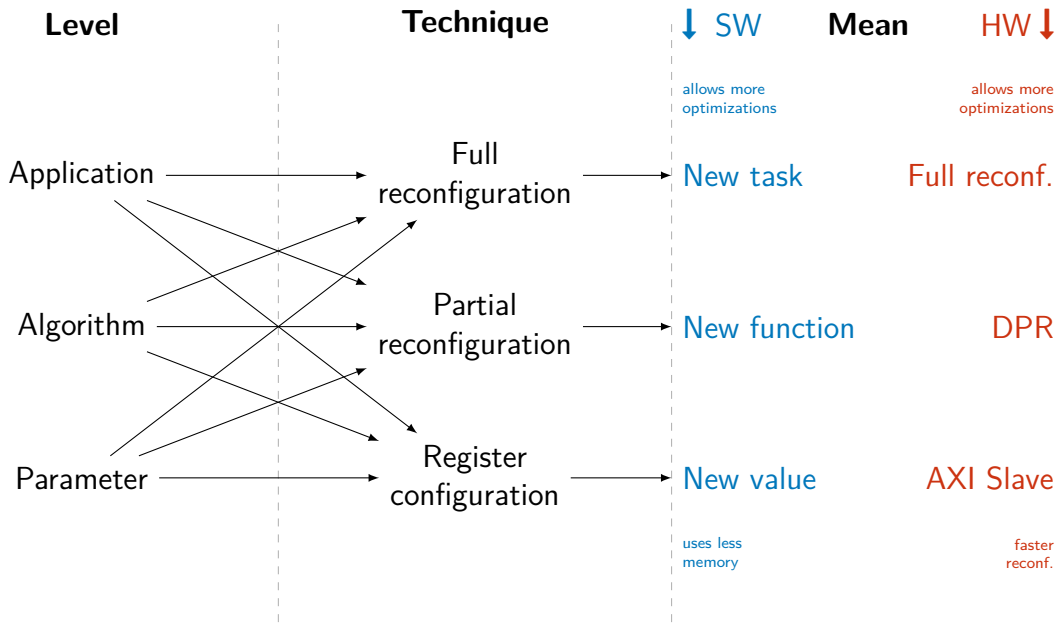


Figure 1.10 – Dynamic reconfiguration levels and methods for SoC FPGA

Reason

The rationales for reconfiguration schemes in radar systems are manifold. First, the most common reconfiguration occurring in a radar processing chain is based on context changes. The simplest example is the PRF. When a target is far away, the radar will

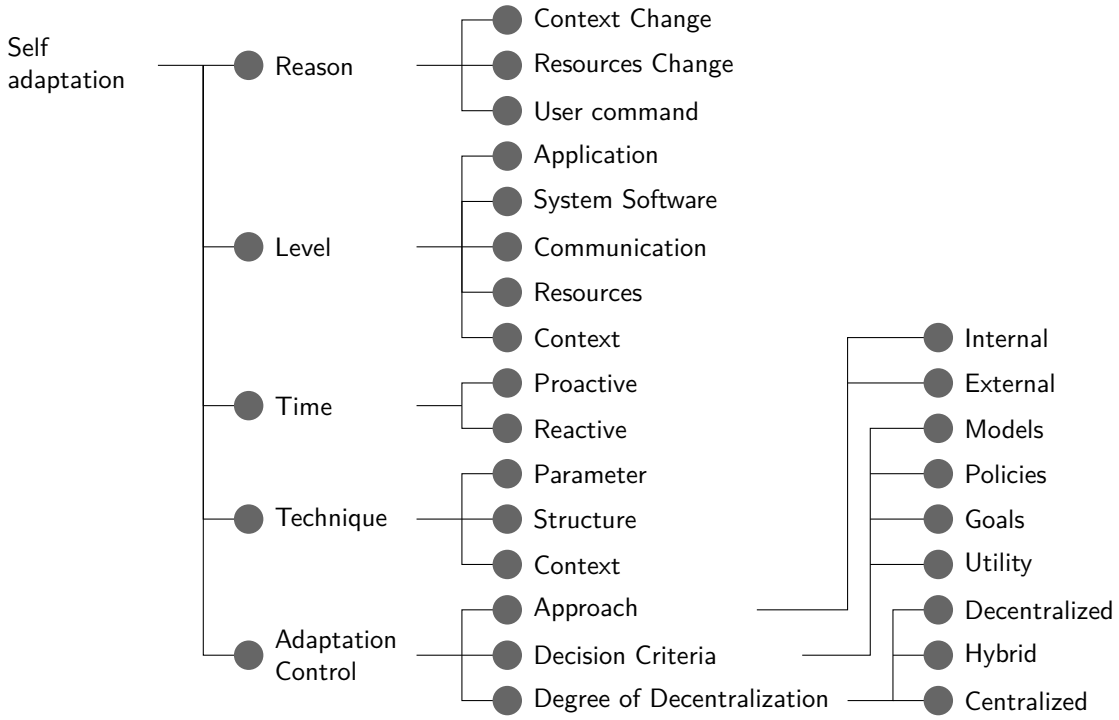


Figure 1.11 – Taxonomy of self-adaptation (adapted from [16] and [17])

reduce its PRF to increase T_r and thus the maximum unambiguous range [2]. As this target gets closer, the radar will increase PRF. Radar systems may have to modify configuration because of resources change, for example in the case of a transceiver malfunction. Finally, another reconfiguration source is based on the user, or more generally on a master system. For example, the master can force the radar system to focus on one particular target and to revise its algorithms accordingly.

Level

Radar processing is an interesting case study for reconfigurable computing. Since this system requires software and hardware, the reconfiguration can happen at many levels.

- The executing application can drastically change. Indeed, modern radar antennas (AESA, MIMO) support different modes to perform, for example, SAR algorithms or tracking (range / Doppler extraction) depending on the context. In this case, a large part of the system will be reconfigured: the system software, the hardware

2. The longest range for which the target echo is received between the corresponding pulse and the next one, equal to: $\frac{c(T_r-\tau)}{2}$.

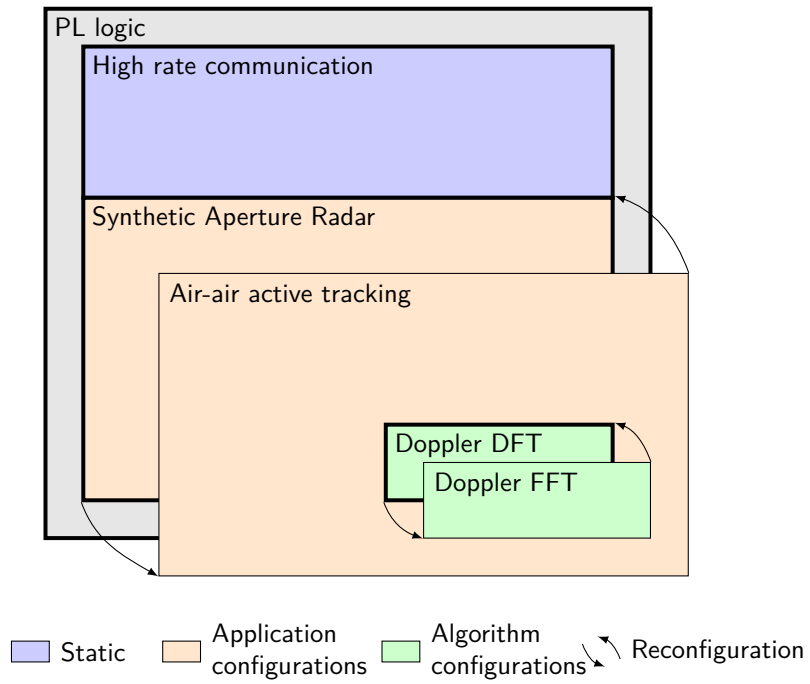


Figure 1.12 – Example of an aircraft radar reconfigurable FPGA architecture

resources (through DPR) as well as the communication means.

- Reconfiguration can also happen at the algorithmic level. In tracking systems, direction finding algorithms and ATR are useless if no target is detected. The resources should be first allocated to enhance detection and then reallocated to these algorithms once detection is successful. In this case, a smaller part of the resources will be reconfigured in SW or HW, but the communication means should not change.

Figure 1.12 depicts an example of a radar FPGA architecture with two level of reconfiguration: applicative and algorithmic. We see on this figure that the multiplicity of configuration levels involves hierarchical configurations. In this example, a reconfigurable algorithm is included in a reconfigurable application.

Time

Most of the time, radar reconfiguration arise in reaction to an event like a context change or a user command. Therefore, reconfiguration arise in reaction to an event. However, proactive reconfiguration is possible in some cases. For example, if a radar detects a target and this target seeks to hide, it can try to disappear in the clutter. The radar can anticipate this behavior by looking at the target trajectory and proact by adapting the

algorithms to be more efficient in presence of a clutter.

Technique

Since the radar reconfiguration can happen on many levels, the techniques used for reconfiguration can take many forms. The faster reconfiguration technique, either for hardware or software is based on parameter changes. The most common example in radar systems is the PRF. However, this type of reconfiguration only allows minor changes. When a whole algorithm must be adapted, the implemented structure is affected. The designer has two choices. Either the different algorithms are implemented and we switch between them with a parameter, or we replace the whole structure. The former is faster in hardware but uses more logic. In software, it can be slower because it requires additional logical tests. The later results in faster execution in every case, but the reconfiguration process takes time in hardware (since it is done by DPR).

Adaptation control

In our example, the adaptation control must be knowledge-based. This kind of control system can be based on the Monitor, Analyse, Plan and Execute (based on Knowledge, a.k.a. MAPE-K) principle. Figure 1.13 shows an adaptation control system managing a radar configuration. There is a hierarchical relation between a slave (radar) and a master (e.g. an aircraft system). The master system has a global view on the mission and a strategic vision. Its goal is to optimize the system to the mission. The master has its own adaptation system, whose decisions directly affects the slave system. The slave adaptation system transmits information to the master, either directly or after transformations, to improve reconfiguration decisions. The slave system also locally adapts by optimizing its configuration according to the master demands (e.g. the direction to observe, the target to track). The radar adaptation control system is therefore both internal and external. The degree of centralization of this adaptation is hybrid. However, even if both adaptation control can take its own decisions, if one of the two adaptation control encounter a failure, the whole system optimization is compromised. This can be mitigated if the master and / or slave MAPE-K is duplicated. In this hierarchical MAPE-K arrangement, the knowledge of the radar slave is limited to its local vision and is shared with the master to enhance its global vision on the mission.

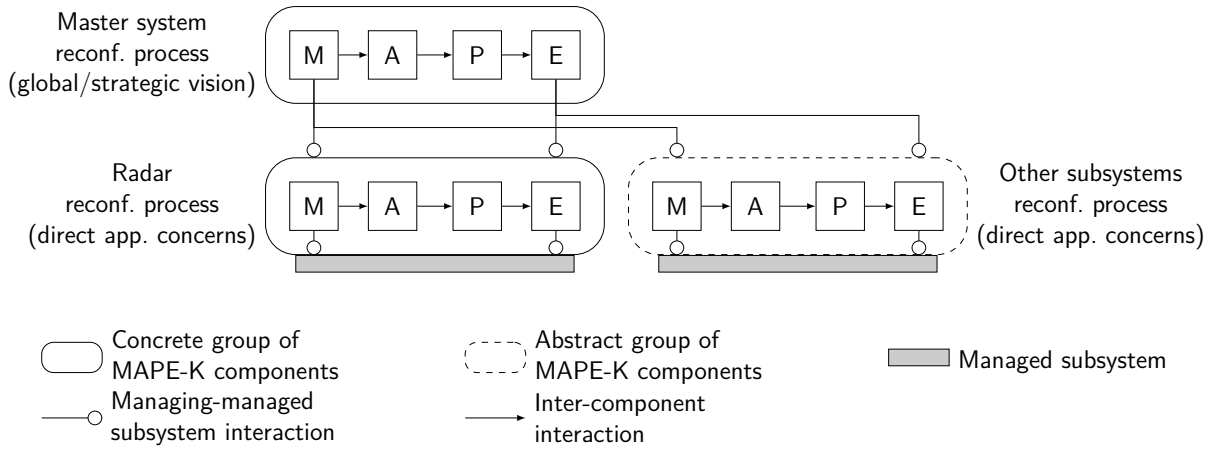


Figure 1.13 – Hierarchical MAPE-K process of a reconfigurable radar system (adapted from the hierarchical model from [18])

1.2.2 Common tools for Dynamic Partial Reconfiguration (DPR)

The dynamic partial reconfiguration requires offline and online operations. The implementation of the computing architecture on the FPGA and the different configurations that must be used by the system is performed offline. The system embeds the bitstreams of the different configurations in memories and must control this reconfiguration through HW and/or SW online.

Implementation workflow

The implementation of HW reconfigurable architecture is supported by several tools from the main FPGA vendors (Intel and Xilinx). The workflow is similar and is based on a hierarchical implementation. Its is composed of the six steps depicted in Fig. 1.14:

1. Logic synthesis of the hardware architecture (block-based transformation of Hardware Description Language (HDL) code to netlists).
2. Reconfigurable partitions (RP) definition in the reconfigurable logic and association of the reconfigurable netlists to RP.
3. Placement and routing of the architecture with one configuration in each RP and writing of the bitstreams.
4. Replacement of the RP by black-boxes.
5. Locking of the static architecture placement and routes.

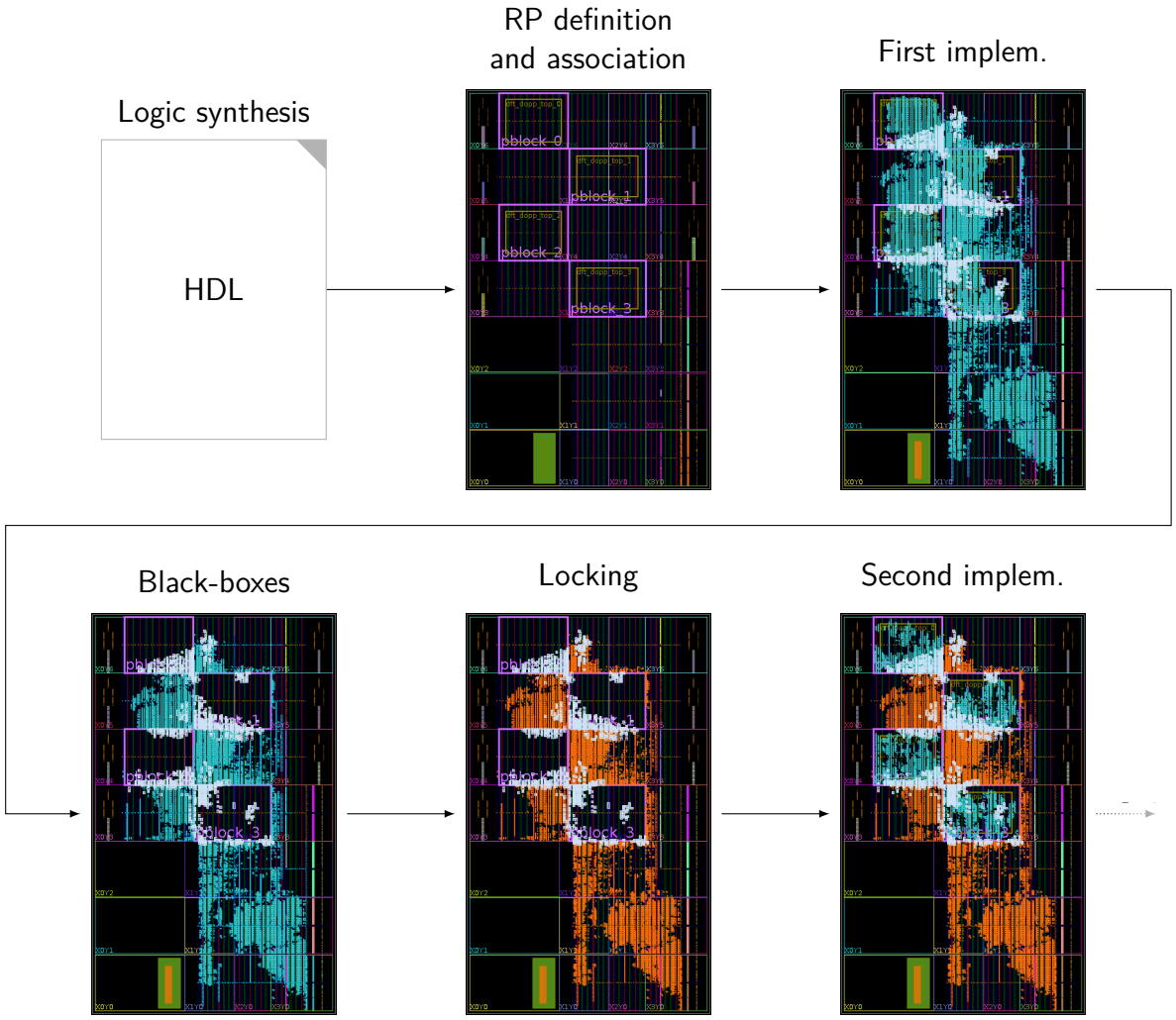


Figure 1.14 – Example of a reconfigurable system implementation with Vivado

6. Placement and routing of the remaining configurations in place of the black-boxes and writing of the bitstreams.

This workflow is straightforward and well integrated in the vendors design tools with many provided tutorials. However, this simple workflow does not allow some optimizations, for example hierarchical reconfiguration [3] (reconfiguration of a partition which is a subset of another partition, which would be required to implement the architecture of Fig. 1.12). Some academic tools provide frameworks to allow more optimizations (see Chapter 2) but are usually also less robust and up to date.

3. Intel FPGA supports hierarchical reconfiguration with a slightly different workflow, see [19]. At the time of writing, Xilinx still does not support this kind of implementation.

Reconfiguration control

The control of DPR is composed of HW and SW parts. In HW, the designer must implement a device interfaced to the configuration port to perform DPR. This is not required in the ZynQ SoC if the reconfiguration is controlled from software since a dedicated device, the Processor Configuration Access Port (PCAP), is available. The designer must also implement logic to freeze the communications on the extent of the reconfigurable logic during the reconfiguration time.

The vendors provide configurable IPs to implement the reconfiguration device and the freeze functions. The FPGA vendors provide APIs to control the DPR from software. This is convenient for SoC platforms such as the Zynq[20] and the Arria[21] SoC FPGAs. However, these API only allow to perform a standard reconfiguration and as explained afore, in embedded systems we might have to perform several reconfigurations depending on context changes. A critical task is to gather information on the mission context to adequately adapt the system configuration. This requires integration of application expert knowledge in the reconfiguration control (e.g. through a MAPE-K control) and this is not supported by standard tools since they are designed for HW and SW experts.

1.3 Conclusion: radar reconfigurability

Table 1.1 lists configurations examples of an hypothetical aircraft AESA radar. In practice, some configurations are selected at design time, and implemented side-by-side. Several works have proposed reconfigurable radar architectures [22], [23], but these reconfigurable systems are based on an external command to take the reconfiguration decision. It might be viable for some types of radar reconfiguration (e.g. from tracking to communication) but has limitations.

System automatic reconfiguration must be based on criteria to take the right decisions at the right moments. While for a lot of systems it is possible to rely on well understood criteria (e.g. power consumption, resources availability and execution time), airborne radar systems require more subtle criteria in order to understand the system situation and environment. Table 1.2 presents criteria examples which can be used in radar to monitor the mission context and to measure the system QoS. We observe that these criteria are far from the more usual ones of power consumption and speed of execution used in many reconfigurable systems.

Identifying reconfiguration opportunities and criteria is not an easy task and requires

application expertise. For example, if we imagine a system with active or passive tracking implementation when the target uses a jammer. How can the system determine that the jamming is too strong and requires to use passive tracking to have better performances than active tracking? In addition, how to be sure that the jammer effectively comes from the target and not from another object? This example gives an insight on the importance to have QoS indicators of the radar processing chain to help the reconfiguration choices. Examples of reconfigurable radar architecture developed in this thesis will give more precision on the forms that QoS-based reconfiguration of radar systems can take. We chose to work on three reconfigurations examples, which are developed in Chapters 3 and 4, to demonstrate the importance of reconfiguration in airborne radars:

- The tracking algorithm with an algorithmic reconfiguration based on the innovation likelihood criteria,
- The Doppler extraction with an algorithmic reconfiguration based on the PDAF validation criteria,
- STAP dimensions with an parameter-based reconfiguration based on machine learning prediction.

In this chapter, we presented an overview of the diversity of radar algorithms and the benefit that reconfiguration can offer to modern radar systems. We will now present a state of the art focused on the technologies and processing studied in depth throughout this thesis.

Table 1.1 – Radar configurations (non-exhaustive list for airborne AESA)

Configuration	**Level**	**Description**
Pulsed[1,2,3]	Application	Uses a chirp waveform (long range).
CW[1,2,3]	Application	Uses a continuous waveform (short range if their is no need to know the range).
FMCW[1,2,3]	Application	Uses a frequency modulated continuous waveform (short range or target-clutter discrimination).
Air-air active search-and-track[1,2,3]	Application	Performs air-air detection and tracking with pulse or FMCW.
Air-air passive $\lambda : cm \rightarrow m$ search-and-track[3]	Application	Performs air-air detection and tracking from jamming signal or target radar emission. (time reversal method?)
Air-air passive $\lambda : mm$ search-and-track[3]	Application	Performs air-air detection and tracking from the thermal signature of target against background
Air-ground Ground ranging[1]	Application	Uses air-ground steering to estimate ground variations and allows very-low-flying.
Air-ground SAR[1,2,3]	Application	Performs air-ground detection through radar imaging.
High rate communications[6]	Application	Uses the antenna to communicate (e.g. for identification or cooperation).
Constant velocity[2] tracking	Algorithm	The Kalman model considers that the target velocity is constant.
Constant acceleration[2] tracking	Algorithm	The Kalman model considers that the target acceleration is constant.
Highly maneuvering[2] target tracking	Algorithm	The Kalman model considers a highly maneuvering target (can require IMM).
Doppler DFT[5]	Algorithm	Uses DFT to perform Doppler extraction.
Doppler FFT[5]	Algorithm	Uses FFT to perform Doppler extraction.
Antenna online auto-calibration[4]	Algorithm	Solves direction of arrival equations to deduce antenna characteristics (e.g. based on RARE and MUSIC).
HPRF[1,2]	Parameter	High pulse repetition frequency (short range).
MPRF[1,2]	Parameter	Mean pulse repetition frequency (mean or unknown range).
LPRF[1,2]	Parameter	Low pulse repetition frequency (long range).
Range-Doppler selection[5]	Parameter	Selection of the range-Doppler map window to observe.
Coherent / noncoherent integration[2]	Parameter	Selects the number of CPI to integrate. (can be high to improve SNR, or low for better reactivity).
Reverse engineering countermeasures	Other	Modify the implementation to make reverse engineering more difficult.

Sources: [1][24]; [2][25]; [3] [26]; [4] [27]; [5] [3]; [6] [28], [29]

Table 1.2 – Radar QoS criteria

Criterion	Level	Source	Description
Target property change	Context	Algorithmic	The target changed its range, speed and/or trajectory.
Kalman Innovation likelihood	Context	Algorithmic	Coherency between measurement and tracking model.
PDAF validation	Context	Algorithmic	The target is associated to a valid track. Derived from the innovation likelihood.
Clutter identification	Context	Algorithmic	Identification of clutter in the signal.
Jamming identification	Context	Algorithmic	Identification of a jamming source.
Antenna fault	Resources	Algorithmic or hardware	Antenna performances degradation or an antenna element failure.
Calibration uncertainty	Resources	Algorithmic	DoA difference between robust and non-robust algorithm or DoA likelihood in Kalman filter.
Radar fault	Resources	Algorithmic or hardware	Fault in the radar processing system.
Radome attenuation	Resources	Algorithmic or hardware	The radome is wet so the antenna performance is reduced.
Attack detection	Resources	Algorithmic or hardware	An active attack is detected by hardware (either fixed or configured on FPGA).
Change focused target	User	External	The radar is expected to track one or several specific target.

Chapter 2

State of the art

Contents

This chapter present a state of the art composed of three sections. The first section is dedicated to dynamic partial reconfiguration with a focus on radar systems. The second section is devoted to the three algorithms studied in our reconfiguration case-studies. The last section presents a state of the art of methodologies and frameworks available to design HW/SW reconfigurable systems.

2.1 Dynamic partial reconfiguration

DPR allows runtime adaptation while exploiting FPGA features, such as speedup and energy efficiency. Indeed, for target domains well suited for FPGA that benefit from parallel computation, bit-wise operations and fixed-point arithmetic, DPR provides a more efficient solution than GPU and multi-CPU while ensuring flexibility [30]. Moreover, a reconfigurable FPGA-based system allows the use of smaller chips, which offers a cost advantage over an equivalent multi-board solution [22]. The flexibility provided by DPR can be leveraged in different ways detailed in the next section.

2.1.1 DPR use-cases

As mentioned in [31], the reconfigurable systems discussed in the literature serve different purposes, such as application acceleration, reliability improvement and adaptivity:

- **Application acceleration by scheduling:** some examples handle reconfigurable modules (RM) like tasks. In [32], the authors use DPR to implement computing tasks on the FPGA, in a sequential order given by the application. Figure 2.1 illustrates the advantage of DPR-based scheduling approach (at the bottom) versus the equivalent static implementation (at the top) which requires a larger FPGA. In this work, the software is bare machine and the scheduling is static, i.e. defined offline by a finite state machine such as the one at the top right of Fig. 2.1. In [33], the authors propose an OS for real-time tasks online scheduling on HW. This type of system is meant to reduce the global power cost and increase performance using FPGA. In this case, the reconfiguration decision only depends on the software system scheduling.
- **Self-repairing:** some applications use DPR to enable HW logic self-repairing. To do so, two techniques are used. The first is based on scrubbing and consists in periodically rewriting the critical reconfigurable partitions configuration, to ensure that

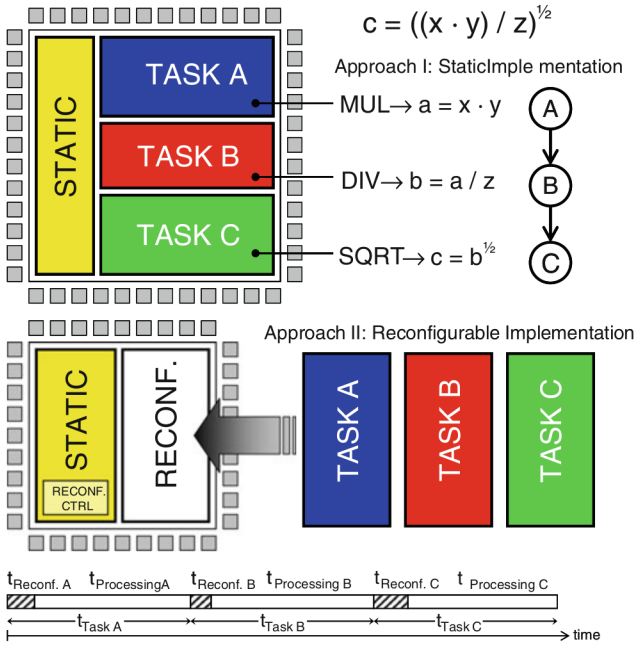


Figure 2.1 – Comparison of static implementation and static scheduling (Fig. 1 from [32])

this configuration is valid. This method downside is that we lose time reconfiguring even if the logic is not compromised. The second, based on fault detection with HW logic duplication or triplication[34], [35] [1], reconfigures if the replicas outputs differ. At this end, [34] presents a tool to automatically implement triple module redundancy at the most critical sections of a FPGA design, for efficient fault detection and configuration scrubbing. Fig. 2.2 presents the case of duplication-based self-repairing, where the output of duplicated modules are compared. If these outputs are equal, we keep the configuration unchanged. Otherwise, it is impossible to know which partition is faulty, so we have to rewrite both. In the triplication case, it is likely that only one faulty partition appears so we know that the replica which outputs a different result is faulty, so we can reconfigure only this partition. In [37] the redundancy-based self-repairing is extended to multi-FPGA platforms.

- **QoS maximization:** DPR may be used to improve the system QoS. In [38], an approach that considers application QoS is described. In this work, the system embeds several configurations from an offline design space exploration. These configurations are associated to their resources consumption, execution time and rules which tell

1. Although it is possible to detect a certain range of defects without duplication. E.g. in [36], authors use the digital root to detect computation faults.

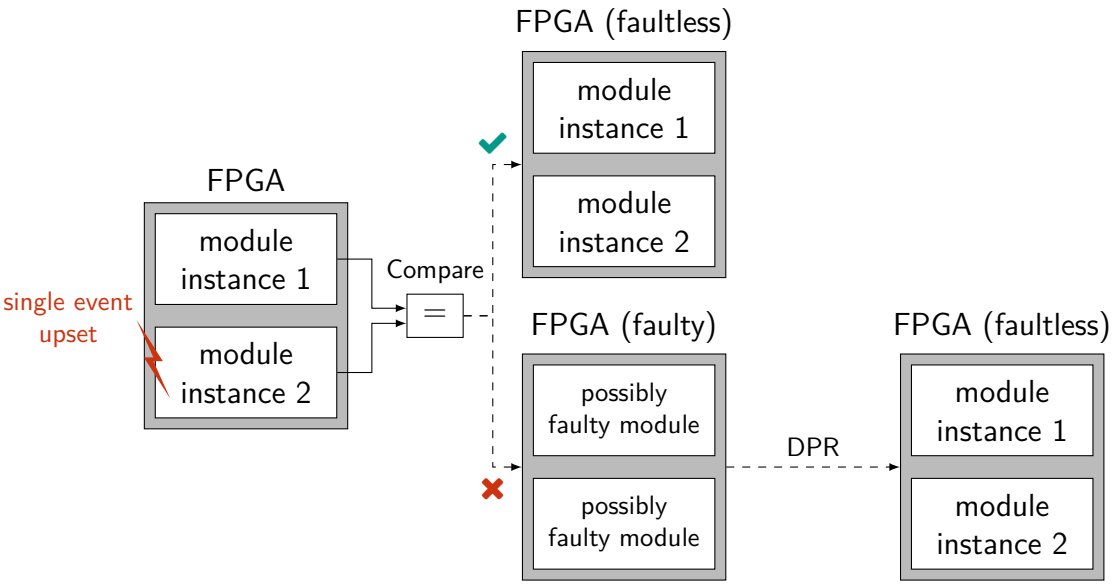


Figure 2.2 – Fault recovery based on a duplicated reconfigurable module

the configurations impact on the QoS. The system then adapt its allocation online from this information. Nevertheless, this work assumes that the system knows every configuration impact on the QoS. In real-life applications this assumption is not always possible. Anyway, the DPR is simulated but not implemented in this work. In [39], authors have designed a system which decides whether to reconfigure according to diagnostic indicators that include application QoS values. The decision method is based on Markov decision process and Bayesian networks, and consider available QoS criteria. However, once again, this work doesn't fully demonstrate the full DPR implementation and suppose available QoS metrics and policies. The real-time operating system proposed in [40] adapts its configuration according to QoS objectives, the position prediction errors in the case study. In [41], authors describe a reconfigurable architecture that replaces HW accelerators to provide required functions to a network infrastructure. This architecture uses a function list to be implemented as well as CPU time and FPGA available space to coordinate the resources allocation. Since the decision is based on the current network state, this coordination leads to the network QoS improvement. In [42] a QoS-driven reconfigurable system uses the application heartbeat to select the best implementation at runtime. Figure 2.3 illustrates the effect of the reconfiguration control implemented in [42] on the QoS (the heartbeat). We can see that, when the QoS curve dives below the defined limit

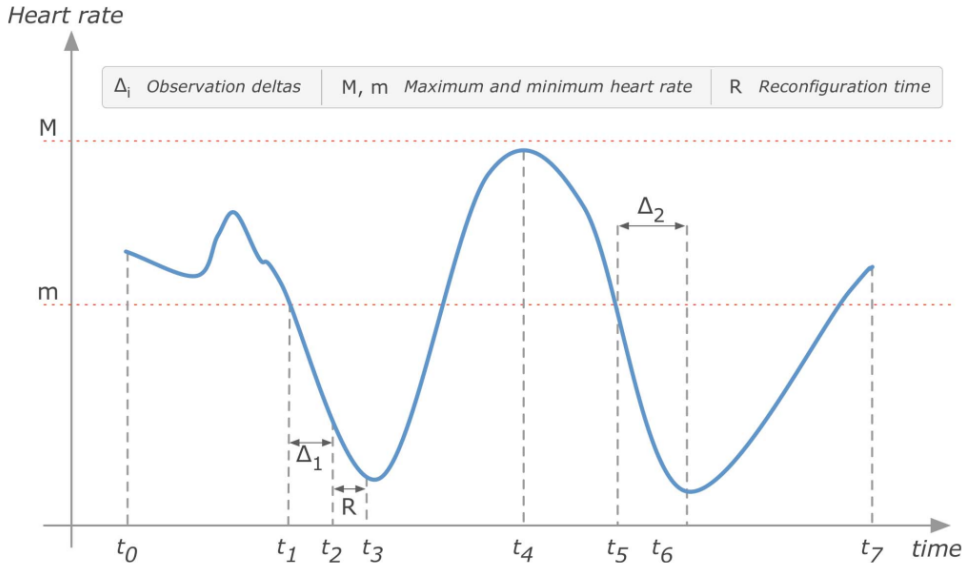


Figure 2.3 – QoS improvement based on reconfiguration (Fig. 4 from [42])

m, the system use DPR to implement a more adapted configuration so the QoS improves.

Generally speaking, architecture optimizations based on HW reconfiguration are mainly considered to enhance performances and reduce power consumption [32], [41], [43]. However, the best HW/SW configuration must first meet the functional requirements which can depend on the operating context. Within this uncertain environment, the choice of used functions and algorithms may change. It results from a large design space and depends on the environment and the execution condition (e.g. mission hazards for an autonomous system). So the configuration choice must be related to the QoS indicators evaluation. QoS designates a measurement of either the system performance or the algorithm-scenario matching. As previously stated, our study focuses on airborne radars, which are confronted with a constantly changing environment. Hence, in this thesis, we focused on QoS-based reconfiguration which allows to analyze the current performance and the system situation to adapt accordingly.

2.1.2 Radar systems DPR

Few works have addressed the question of improving radar systems from DPR [22], [23], [44]. In [22], [23], authors propose to use DPR to switch between radar modes. In [22], the radar system uses DPR to modify the digital beamforming accelerator to perform

aircraft or weather tracking, which require a different number of beams and operating frequencies. In [23], the radar is able to perform tracking or communication by replacing the signal processing module with DPR. These are good examples of DPR use to adapt to different situations. However, in [22], [23], the reconfiguration controls are external and not based on any QoS indicator. In [44], authors optimize a radar application from a simple criterion, which is the distance to the target. In this study, the optimization does not consist in switching between optimal or degraded algorithm but instead choosing the most adapted algorithm depending on the current situation.

We observe from the literature that the reconfiguration criteria are diversified and domain-dependent, and that only application experts can choose and formalize them. In this thesis, we chose to explore three case-studies to expose the radar reconfigurable processing possibilities and challenges.

2.2 Reconfigured application

We identified three reconfiguration examples which can benefit to UAV/airborne radar systems. The following sections present a state of the art for these algorithms.

2.2.1 Kalman filter

Kalman filters are widely used for tracking in systems equipped with various sensors such as cameras (e.g. [45], [46]) or radars (e.g. [47], [48]). These filters, based on a system modeling, allow to compute an object trajectory and its evolution through time. For a unique system, there may be several ways to model trajectories, with more or less success and computing complexity.

In radar systems, we observe different kind of targets, hence various motion dynamics. Moreover, a single target may radically change its trajectory. In these situations, there is not one efficient model but several ones, and it can be interesting to run several models to enhance the tracking. We can simply run several models separately and identify which one performs the best (see [49], section IV), by observing the KF innovation likelihood which estimates the model-observation adequacy. For examples, Figure 2.4 shows the tracking output of Kalman filters on different models. The input trajectories at the left and right side of the figure are created from 'Model 1' and 'Model 2', respectively. A noise is added to the trajectories to create measurements, which are filtered by two KF based on the

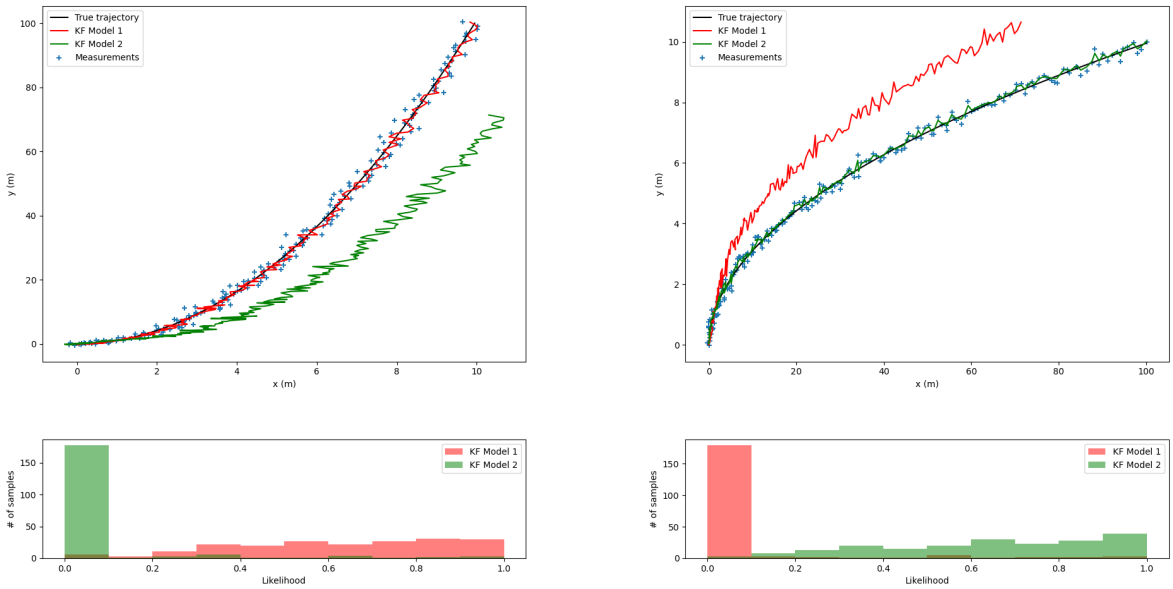


Figure 2.4 – Example of two trajectories and tracking results using Kalman filters (at the top), and histograms of innovation likelihood (at the bottom). The trajectory are generated from 'Model 1' and 'Model 2' at the left and the right of the figure, respectively.

two models. We see for each tracking plots that when the KF model is well adapted to the real trajectory, the filtering is more accurate. In addition, the innovation likelihood histogram is presented for the two trajectories at the bottom of the figure. We observe that the likelihood of the filter is close to zero when the model is not adapted to the measurement, while it is mostly greater than 0.5 when it is adapted. Consequently, it is possible to know which filter performs the best.

A better solution is to use an interacting multiple model (IMM) Kalman filter [50] which runs several models and recombines their state estimates.

However, running a single Kalman filter model in embedded radars is already highly computing intensive, since its complexity grows exponentially as the state space increases linearly. A multiple-model Kalman filter requires more models, depending on the diversity of targets we need to deal with. If we consider a SoC FPGA platform, it is possible to implement two or three Kalman filters in SW and in HW as accelerators. However, with only few models implemented, we must make specific assumptions about the target to track. If these first assumptions are wrong, we should be able to adapt the model during the mission. In SW it is straightforward but for HW, no literature example demonstrates the Kalman filters adaptation in HW from a performance criterion. The closest example in the literature is presented in [44], where the system adapts the Kalman model based

on the distance to targets. However, the criterion used in this example is not adapted to airborne radar tracking, since it is not based on the targets trajectory models.

2.2.2 Discrete Fourier transform

In radar tracking systems, the environment is composed of targets, clutter and other perturbations, which can be characterized by computing a radar signal range-Doppler map (i.e. a discrete version of the time-frequency analysis first described in [51]). In practice, this function is composed of a matched filter and a series of Fourier transforms. This transform can be performed with different methods, as described in [52]. The main methods are the FFT (see [53], [54] for implementation examples) and the classic DFT summation (see [55], [56] for implementation examples for a passive and a pulse-Doppler radar, respectively). The two approaches present different characteristics in terms of throughput, latency, processing and memory resources, that can benefit to different phases of application scenarios. In the multi-channel radars context, the range-Doppler function becomes a major concern [57]. Indeed, although the AESA spatial degrees of freedom give extra information on the targets, this function must be computed for every channel, consuming a lot of logic resources. In this context, the DFT method used for the range-Doppler transform is fixed at design time. Hence, the system cannot benefit from different DFT architectures. DPR, which is proved to be particularly efficient in the signal processing field [58]–[60], permits to overcome this limitation.

2.2.3 Reduced dimensions STAP

The signal measured with a radar is composed of target returns, noise and perturbations. The power of jammer and clutter can be times stronger than the sought signal. AESA are composed of many antennas, allowing the use of Space-Time Adaptive Processing (STAP)[61]. STAP discriminates interferences based on correlations in the space-Doppler domain. This two-dimensional filter offers considerable signal to interference plus noise ratio improvements, the huge computational complexity and memory usage are prohibitive. Full STAP algorithms (i.e. with the maximal dimensions allowed by the radar) include the estimation and inversion of a NM matrix for every range cell, with N the number of antenna elements and M the number of pulses. In addition, several drawbacks exist to the full STAP implementation. Firstly, the number of samples required to estimate the covariance matrix must be equal or greater than NM (ideally $2NM$). Secondly,

larger dimensions of a matrix contribute to the condition number increase which is an indicator of the computing errors which can occur on matrix inversion. In practice, experts use reduced STAP algorithms [62]–[66], for a tradeoff between quality of results and computing efficiency. Two methods can be employed:

- Fixed reduced dimensions: N and M are fixed at design time,
- Adaptive dimensions: N and M are computed at runtime, e.g. from eigenvalues. This subject to computational round-off errors (see [67] for a discussion on random eigenvalues of large covariance matrices).

Optimally, the STAP dimensions adaptation should depend on the interferences correlations in the space-time domain. Jammer and ground clutter, exhibit different properties; hence they have various space-time correlations. Since these correlations are *a priori* unknown, designers usually fix empiric dimensions (e.g. $N = M$). In some situations, we could further reduce the STAP filter dimensions by predicting the most appropriate N and M values at runtime. For instance, a broadband jammer can be effectively filtered out using only the spatial dimension ($N > 1$, $M = 1$). In contrast, a clutter seen from a low speed aircraft will be better filtered with $M > N$.

Several authors [68]–[70] proposed approaches for the STAP dimensions dynamic allocation. The approach proposed in [68] requires performing STAP with the maximum number of cells of one dimension for both STAP dimensions and computing the obtained residual noise levels ratio. The solution presented in [70] uses an iterative algorithm which performs a series of channels tests/selections to select the most relevant antenna elements and backscattering pulses among the N/M dimensions. The dimension selection from these algorithms is computing intensive.

2.3 Methodology and tools for reconfigurable systems design

2.3.1 DPR computer aided design

Since HW systems are still more difficult and costly to develop than SW ones, special academic efforts are made to create CAD tools to ease the design process. DPR is no exception, and many tools were proposed to ease the reconfigurable systems implementation [31]. These tools enable the DPR description from low level [71], [72] up to

C++/SystemC level [73]. Some tools allow to implement and replace OpenCL kernels thanks to DPR [74]. However, the description of efficient HW accelerators with C++ or SystemC already imposes specialized code which requires good HW skills, far from the usual knowledge of application experts [75]. With the DPR additional complexity layer, we see that these efforts do not allow an application expert to efficiently benefit from reconfigurable FPGA.

Designers could benefit from a framework which eases the reconfigurable systems design by different experts (specialized in application, HW or SW). Creating such design framework can be a difficult task with many considerations. In order to define the methodology and the associated tools, we need a way to model how the agents collaborate to create an efficient design. Since there are similarities between cooperative design and multi-agent systems, the design framework could be inspired by the multi-agent domain.

2.3.2 Multi-agent development framework

Literature provides many definition of the agent concept[76], [77]. As there is no standard agent definition, we select characteristics which are commonly attributed to the agents and which can support our methodology representation. The agent most important characteristics are that it is self-organized, cooperative, communicative, has a specific skill set, is reactive and proactive.

For J. Ferber[76], there are several reasons to create a multi-agent problem representation:

- "The problem is physically distributed": the agents are at a different place. For human agents, this is particularly true with the recent teleworking development.
- "The problem is functionally distributed": it requires various knowledge and points of view.
- "The problem is too complex and vast to be completely analysed by one agent": SoC are complex systems which mix HW, SW and application considerations.
- "The system must adapt to structural changes": Some agents can appear, disappear or be replaced without compromising the system. In addition, an increasing number of tasks will be performed by automated processes and AI, which will either increase the number of agents or replace human agents.

This corresponds to some of the problems posed by a SoC design methodology. Multi-agent systems have already been used for the multi-disciplinary design of complex systems.

However, these approaches are often too conceptual and generic and lack the specifications to be applied to a particular application area, as in [78]. It is, however, possible to draw elements of response from the work of Y. Chen et al. [78], such as the driving of solution by the functional requirements, the need for clear functional knowledge representations, the management of knowledge and the sharing of a common solutions pool.

2.4 Positioning of the thesis work

In previous works DPR has been widely used. However, most of them focus on changing implemented logic based on application scheduling or external commands. Several works focus on maximizing application QoS, but only a few studies address the case of embedded radar systems. In order to further explore the possibilities of embedded radar systems dynamic reconfiguration, this thesis presents three original reconfigurable radar algorithms case-studies:

- In Section 3.1, we propose a reconfigurable architecture for Kalman filters which modify the implemented models, using innovation likelihood as a QoS criterion.
- In Section 3.2, we propose two contributions for the discrete Fourier transform in the context of Doppler radar:
  - ▷ A reconfigurable architecture which allows to use the proper version of DFT algorithms according to application needs, without the resource consumption overhead. This architecture limits the impact of the reconfiguration time by performing an original progressive reconfiguration that opens new perspectives,
  - ▷ A method to take the reconfiguration decision at the right time in a radar application.
- In Chapter 4, we propose a lighter method for STAP dimensions reduction based on light neural networks and random forests. Indeed, we came to the conclusion that a machine learning process could extract fine information from the noise plus interferences covariance matrix to adjust the N and M dimensions while the system is running. Thus, we propose STAPLE, a method based on machine learning prediction models to determine the most adapted N and M dimensions of the STAP before performing the costly computation. Our prediction models base the inferences on the interference condensed observation (being the prediction models input) offered by a reduced dimension covariance matrix, this matrix being first estimated in the

online process (and then avoiding the tedious full dimension matrix estimation). The prediction models are based on a combination of light Convolutional Neural Networks (CNN) and Random Forests (RF), which have good characteristics to be implemented within highly parallel and pipelined architectures (e.g. FPGA or GPU implementations[79]–[81]). We compare our approach to the standard $N = M$ approach using the Improvement Factor (IF), matrix condition number and computing complexity as metrics. Our solution holds a lower computing complexity and exhibits more computation parallelism, thus shorter response times.

From the observations made on these case-studies, we then propose a methodology in Chapter 5 to ease the reconfigurable systems design by different experts and using existing tools. This methodology aims to raise the reconfigurable system description level in order to make it accessible to application experts, while letting the implementation considerations to the HW experts. It means that our methodology must be compliant with most flows and tools for the final implementation. Hence, we use a multi-agent representation to formalize the relations between the designer and the development environment towards a common goal: the design of an efficient HW/SW computing core. This formalism allows to foresee the extension of such methodology to include AI and autonomous agents.

Chapter 3

HW Reconfiguration based on Algorithm choice

Contents

This chapter presents two case studies of radar processing hardware reconfiguration based on algorithmic QoS indicators. Firstly, Section 3.1 propose a basic methodology to design a reconfiguration controller based on quality of service (QoS) indicators to be specified by experts. This controller monitors the reconfigurable partitions (RP) and can make DPR decisions to optimize the global QoS. We illustrate our methodology in the Radar domain with a tracking system based on Kalman filters implemented on a Zynq Ultrascale+. This study highlights expected gains and obstacles, it presents the different strategies to cope with the issues and draws perspectives. Secondly, Section 3.2 presents a reconfigurable system which switch between a classic discrete Fourier transform (DFT) sum and a fast Fourier transform (FFT) to enhance Doppler extraction. This section explores the pros and cons of both methods. Based on these observations, we propose a new architecture and decision method that relies on radar QoS for enabling an efficient self-adaptive solution. This method is tested in a hardware-in-loop simulation with a DPR radar implementation. These two studies have been published in the in the context of this thesis, and correspond to references [2] and [3] for sections 3.1 and 3.2, respectively.

3.1 QoS Driven Dynamic Partial Reconfiguration: Tracking Case Study

This work proposes a simple methodology to control QoS-based reconfiguration and demonstrates this methodology with a case-study based on radar tracking.

In section 3.1.1, we present a methodology to create an adaptive reconfigurable system to reconfigure regarding application QoS feedback. The case study of Tracking method, based on Kalman filters and implemented with the proposed QoS driven reconfigurable architecture, is described in section 3.1.2. Section 5.6 concludes the study and draws directions for future work.

3.1.1 Methodology

Separation of concerns

To take advantage of a QoS driven reconfiguration, the signal processing experts must be DPR aware so that they can have in mind the possibility to switch between different algorithm modes or configurations. However, these experts should not have to care about the implementation details. Thus, the design of a QoS driven DPR needs to be based

on the separation of concerns scheme depicted in Figure 3.1. Once the signal processing experts know they can use several HW configurations in one mission, they can explore more ambitious strategies. First, they need to specify the algorithms to implement and the different versions they need, in terms of performances and resources cost. Secondly, they have to define the QoS indicators they need to implement such decisions. The SoC designers then can provide the required feedback in the same way as in a classic development process.

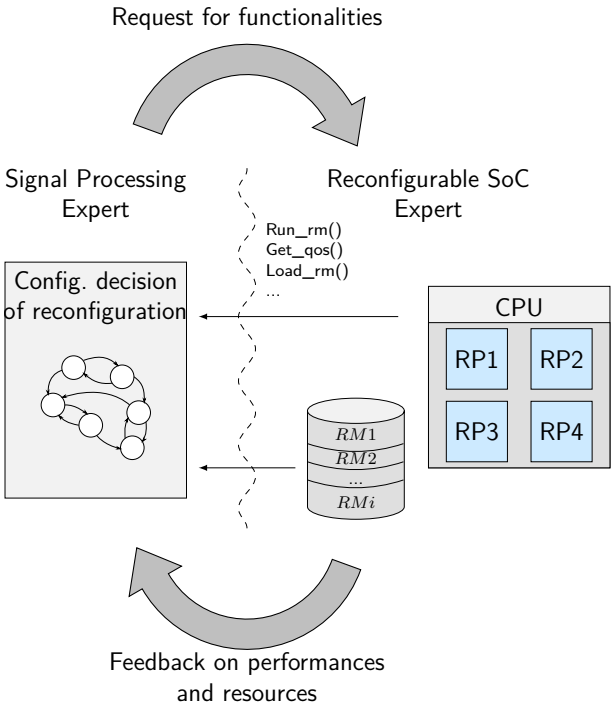


Figure 3.1 – Schematic representation of the separation of concerns in the QoS driven DPR methodology

After a few cycles, the reconfigurable SoC experts have to bring to the signal processing experts a library of the different configurations (RMi in Figure 3.1). They must also provide simple API for the experts to implement the reconfiguration decision system.

Reconfigurable architecture model

The architectures considered in this study target SoC FPGA devices. They are composed of a FPGA (PL) and a processing system (PS). Several partitions are defined to be reconfigurable. The reconfigurable partitions (RP) communicate with one another and with the PS through buses. The more generic the buses are, the more heterogeneous the

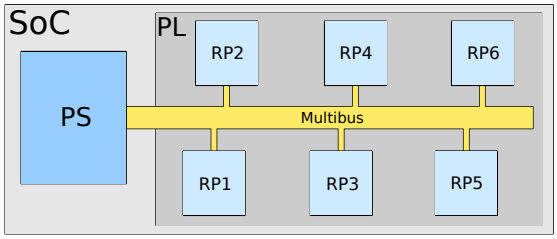


Figure 3.2 – Example of a SoC reconfigurable architecture.

RM can be. The reconfiguration can be triggered either by the PS or by the PL part. The reconfiguration controller must also access the partial configuration bitstreams that can be stored in the main shared DDR memory or in dedicated one. However, a QoS driven methodology must be accessible to algorithm experts who are not familiar with FPGA. So, if the configuration delay is compliant with the application time constraints, a SW implementation is the best solution. As a SW program, the configuration decision and the configuration control can simply be tested and modified by algorithm experts. Figure 3.2 illustrates an example of such an architecture with six RPs implemented on a SoC FPGA.

Reconfiguration control system

Besides the execution flow, the QoS-based controller can be decomposed into four parts: 1) Monitor, 2) Analyze, 3) Plan and 4) Execute. The monitor function is in charge of collecting data and computing at least one QoS indicator per RM. These indicators can be computed directly by the algorithm, for example the error in an automation process. If it requires additional computation (e.g. estimation of the next state), the designer can choose to implement it in either the PL or the PS part. The analyze function must determine if a reconfiguration is needed, with regard to the QoS indicators. The plan function decides when to trigger a reconfiguration, and which configurations must be replaced. Finally, the execute function is in charge of loading the right bitstream at the right time according to application execution flows and architecture constraints. In all cases the decision-making based on the indicator is co-designed with application experts and so is preferably implemented in SW. This allows experts to evaluate different strategies. Figure 3.3 illustrates the DPR control flow, step-by-step.

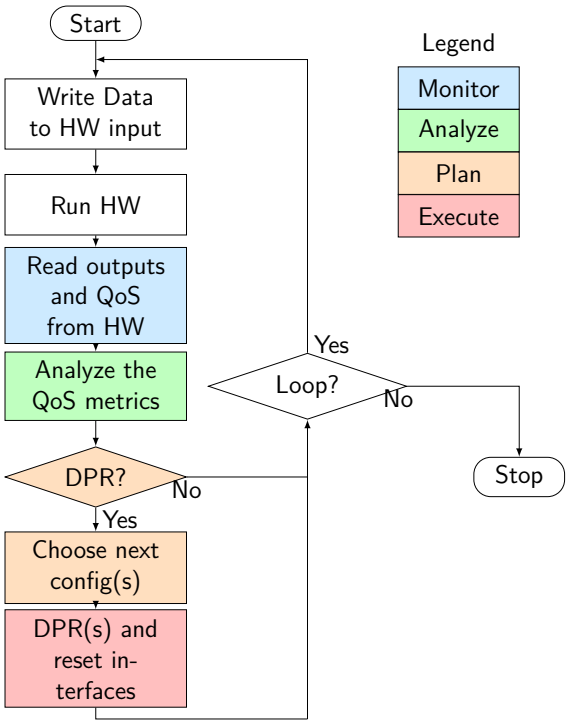


Figure 3.3 – Diagram of the DPR control system

3.1.2 Case study : Kalman-based tracking

We are interested in optimizing the tracking of a target by selecting the best Kalman Filter (KF) chosen in a library of KFs which differ from each other by their process models, i.e., the way they model the trajectory of the target. Ideally we could execute all the models in parallel and select the best one after each iteration according to QoS indicators. However, our major constraint is that all the KFs available in the library cannot be executed simultaneously because of real-time constraints and hardware resource availability. So, only a subset of the library must be implemented. In our study we consider that two KFs can run simultaneously. KFs can be implemented either in the PS or the PL part. The advantage of the HW version is twofold. First, it improves the execution time and so the response time of the system. Secondly, it allows the execution of multiple KFs in parallel while having the PS available for running other application and system tasks. We choose a model with different KFs running in parallel in separate RPs.

In real tracking systems, such Kalman filters can be very large (more than 10 states in the aircraft dynamic navigation filter in [82]). Hardware optimization based on constant matrix elements is then required. Reconfiguration based only on parameter changes is,

therefore, inefficient. Moreover, DPR allows to implement Kalman filters with completely different behaviours (e.g., EKF, UKF and IEKF). In this study, we use simpler Kalman filters to illustrate the method without loss of generality.

Reconfigurable architecture model

The architecture used for the study is based on the one defined in 3.1.1. In this example, two RP are defined. These RP can host a KF implementation. Each RP can be configured with seven different state models, each model corresponding to a different trajectory assumption. This means that they do different computations and use different matrix sizes. The KF gets the measurements from the PS through a direct memory access (DMA) interface. This DMA is included into the RP. The buses use a generic AXI4 interface for PL to PS communications. In this example, there is no communication between both RP.

QoS estimation and DPR decision

The QoS estimator at time k is chosen as the log likelihood of the measurements collected at time k. This criterion can be used as an efficient estimator of the KF model correctness[9]. It can be computed using the innovation and the innovation covariance computed by each KF under test:

$$f_{QoS}(k) = \mathbf{i}_k^{\mathrm{T}} \mathbf{\Sigma}_i^{-1} \mathbf{i}_k \tag{3.1}$$

where:

$$\begin{aligned} f_{QoS} &= \text{QoS function} \\ i &= \text{innovation vector} \\ \mathbf{\Sigma}_i &= \text{innovation covariance} \end{aligned}$$

We observe that this function requires matrix multiplication, so it can benefit from a parallel HW implementation. This is a typical point of useful discussion that authorize the proposed methodology. The QoS indicator is specified by the application designers and the reconfigurable SoC designers can work on the best implementation. In this case, they can reuse the inverse of the innovation covariance matrix which needs to be computed to get the optimal Kalman gain (equation 3.2). Therefore, we chose to implement f_{QoS} in hardware, using Newton division to increase performance while reducing PL footprint.

$$K_k = (P_{k|k-1} H_k^{\mathrm{T}} \mathbf{\Sigma}_i^{-1}) \tag{3.2}$$

where:

$$\begin{aligned} K &= \text{optimal Kalman gain} \\ P_{k|k-1} &= \text{predicted error covariance} \\ H &= \text{observation model} \\ \mathbf{\Sigma}_i &= \text{innovation covariance} \end{aligned}$$

Once this indicator is computed, the IP sends it back to the PS. The application then has access to a QoS value for each of the KF modules. This allows the program to identify which filter is the most adapted to the current situation. In this way, the PS can replace the worst one and implement another IP to test another trajectory hypothesis.

Figure 3.4 shows an illustrative example of the control principle while considering 2 DPR partitions that can implement different Kalman models. The first and second lines draw the evolution of the configurations implemented on the RP1 and RP2 over time.

The method we chose to control the reconfiguration from the QoS values relies on 4 phases:

1. **Initialization:** Two configurations are chosen at the beginning of the tracking. This choice can rely on mission indicators. For example, in radar systems, the target trajectory can be classified with the help of a priori knowledge.
2. **Observation and decision:** QoS-driven adaptation is strongly specific to the application and must therefore be flexible, as a software it can be easily modified according to test results. In our case study different strategies are possible. We have implemented a solution that combines long-term observations and reactivity. The QoS estimator gives a new value at each Kalman call. However, we cannot reconfigure each time we get a new QoS value. Nevertheless, a very bad QoS value means that we need to change the configuration as soon as possible. Therefore, we need to have two kinds of reconfiguration: The first one computes the average QoS over an observation window (shown in Figure 3.4), the KF with the worst QoS score is replaced by a new one according to a round-robin mechanism (Reconf. decision n in Figure 3.4). The second one is based on thresholds, when both KF QoS exceed the maximum values then KF are interrupted before the end of the window and replaced by two new ones (triggered-based reconfiguration decision in Figure 3.4).
3. **Reconfiguration:** The reconfiguration takes time to perform. In some radar applications, there can be no down time. In our system, the reconfiguration time can be hidden since there are two RP running concurrently and only one will be stopped on reconfiguration. The only situation where a timeout occurs is when the threshold-

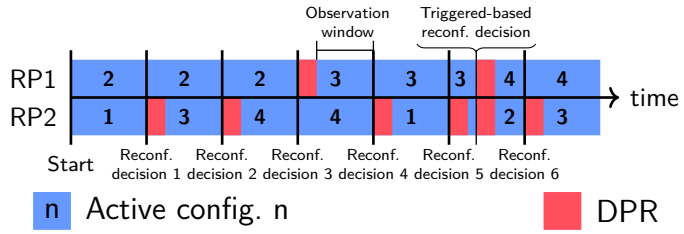


Figure 3.4 – Example of QoS driven DPR control

based reconfiguration is triggered for both RP. In such case, none of the Kalman filters can achieve a good tracking so a HW reconfiguration is required.

4. **Convergence:** When using a Kalman filter, there is a convergence time. This happens because the starting point of the algorithm is not perfect. This time causes the filter to output inaccurate points and can cause the QoS estimator to output bad scores during the convergence interval. In this experiment, we consider this training phase as an additional reconfiguration time. We don't observe the QoS during this phase to avoid penalizing new implemented filters.

Experimental setup

HW/SW platform The reconfiguration strategy as well as the KFs are implemented in a Zynq Ultrascale+, a SoC made up of a multi-core ARM and a FPGA that supports DPR. Figure 3.5 presents the layout of the FPGA, with the two reconfigurable partitions delimited by the purple areas. With this layout, the reconfiguration of a RP takes 8.7ms using PCAP. This time depends on the size of the partial bitstream as well as the bandwidth of the configuration method. The configuration remains static outside these two boxes. In this experiment, we use only one processor core as well as the PL part. In real-life tracking systems, the other cores would likely be running other tasks such as detection.

The different Kalman filters are generated using Vivado HLS, a high-level synthesis tool. This allows to generate different configuration quickly, while ensuring that the different configurations are compatible. Indeed, the RM needs to have the same inputs, outputs and operating frequency. In our tracking example, the target lies on a plane whose axis are x and y, $x \perp y$ (Figure 3.6). The observation vector is $z = [x, y, sx, sy]$, with sx and sy the speeds respectively along the x and y axis.

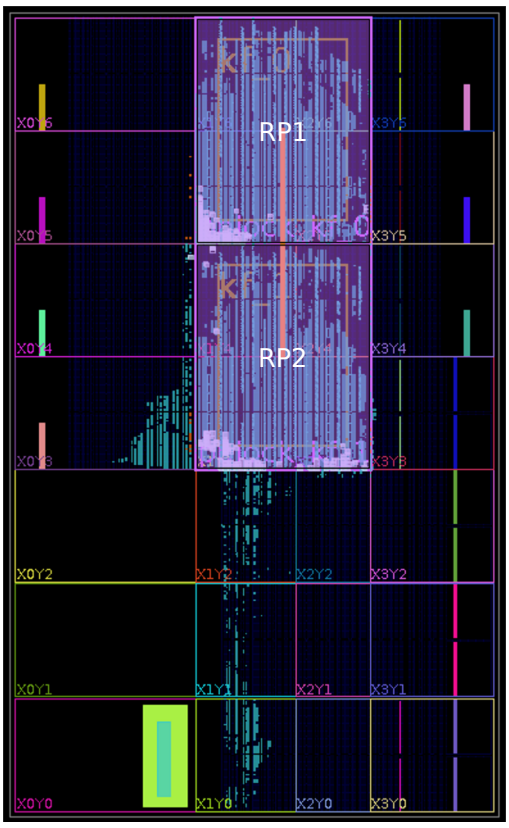


Figure 3.5 – Layout of the FPGA with the two reconfigurable partitions (RP1 and RP2).

KF / EKF models The generated Kalman filters are listed below. They correspond to different trajectories assumptions. Note that a state is considered constant when it is not to be seen in the list.
Linear Kalman filters:

KF-1 : State vector $X_n = [x_n, y_n, sx_n, sy_n]$
State transition equations :

$$\begin{cases} x_n = x_{n-1} + sx_{n-1} * dt \\ y_n = y_{n-1} + sy_{n-1} * dt \end{cases} \tag{3.3}$$

KF-2 : State vector $X_n = [x_n, y_n, sx_n, sy_n, ax_n]$
State transition equations :

$$\begin{cases} x_n = x_{n-1} + sx_{n-1} * dt ; sx_n = sx_{n-1} + ax_{n-1} * dt \\ y_n = y_{n-1} + sy_{n-1} * dt \end{cases} \tag{3.4}$$

KF-3 : State vector $X_n = [x_n, y_n, sx_n, sy_n, ay_n]$
State transition equations :

$$\begin{cases} x_n = x_{n-1} + sx_{n-1} * dt \\ y_n = y_{n-1} + sy_{n-1} * dt \ ; sy_n = sy_{n-1} + ay_{n-1} * dt \end{cases} \tag{3.5}$$

KF-4 : State vector $X_n = [x_n, y_n, sx_n, sy_n, ax_n, ay_n]$
State transition equations :

$$\begin{cases} x_n = x_{n-1} + sx_{n-1} * dt ; sx_n = sx_{n-1} + ax_{n-1} * dt \\ y_n = y_{n-1} + sy_{n-1} * dt \ ; sy_n = sy_{n-1} + ay_{n-1} * dt \end{cases} \tag{3.6}$$

Extended Kalman filters:

EKF-1 : State vector $X_n = [x_n, y_n, sx_n, sy_n]$
State transition equations :

$$\begin{cases} x_n = y_{n-1} * y_{n-1} \\ y_n = y_{n-1} + sy_{n-1} * dt \end{cases} \tag{3.7}$$

EKF-2 : State vector $X_n = [x_n, y_n, sx_n, sy_n]$

State transition equations :

$$\begin{cases} x_n = x_{n-1} + sx_{n-1} * dt \\ y_n = x_{n-1} * x_{n-1} \end{cases} \tag{3.8}$$

EKF-3 : State vector $X_n = [x_n, y_n, sx_n, sy_n]$

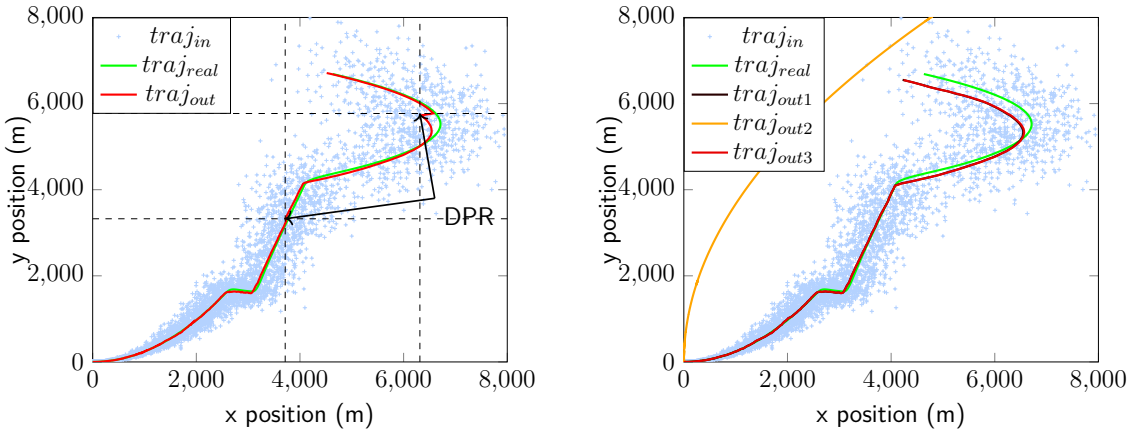
State transition equations :

$$\begin{cases} x_n = \frac{-9.81}{2} * (y_{n-1} - cst)^2 * \frac{1}{(30 * cst)} + sy_0 * sin(\frac{\pi}{2}) * (y_{n-1} - cst) + cst \\ y_n = y_{n-1} + sy_{n-1} * dt \end{cases} \tag{3.9}$$

Benchmark trajectory The trajectory used for the tests is based on a combination of sub-trajectories that follow different Kalman models. In this work, we use a trajectory where the first part corresponds to the equation $y_n = x_n^2$. In the second sub-trajectory, both x and y have constant speed. In the third part, x and y have constant accelerations. The fourth and last part follows the equation of EKF-3. The green curve in Figure 3.6a shows the complete composed trajectory. Finally, the observed values, which are the blue crosses in Figure 3.6a, result from the addition of the target trajectory and a proportional Gaussian noise. These data are processed by the implemented Kalman filters used for the experiments.

Experimental procedure When the designs are synthesized and implemented, the partial bitstreams are loaded in a flash memory. At power-up, the PS loads all the bitstreams in DDR RAM. When a reconfiguration is triggered, the PS feeds the processor configuration access port (PCAP) with the corresponding bitstream.
On power-up, the PS starts a benchmark to test our design. First, the input data is extracted from the SD card and stored in DDR4. Then, the application reconfigures the two RP to set the system into a known state. Depending on the scenario, the two initial configurations may differ. Finally, the processor transmits the data samples and gets the results along with QoS values. The output of the best KF is first stored into DDR4, and then saved to the SD card to be plotted and analyzed.

In our test bed we consider three different scenarios. In the first one, the two initial configurations are the KF-1 and the EKF-3. Since these systems have strongly different behaviours, the system should easily find the best KF. The second scenario starts with EKF-1 and EKF-3. None of these configurations is a good model at the beginning of the



(a) Tracking with DPR (scenario 3) (b) Tracking without DPR (scenarios 1 to 3)

Figure 3.6 – Tracking results

tracking. This scenario can show if the system is able to react fast to a situation where none of the configurations is good and there is a risk to lose the target. The third and last scenario begins with KF-3 and KF-4. As these configurations are close to each other, it will make the configuration choice more challenging.

Results and analysis

These tests involve a reconfigurable architecture with active QoS-driven DPR and the best Kalman without reconfiguration (model described in equations of KF-4). The results obtained with the benchmarks are given in Table 3.1. To compare the different approaches, we measure the likelihood and compute its median. We also collect the maximum and more importantly the minimum measurement of the likelihood, as a small value could cause the system to lose the target. We can observe that for the third scenario, the likelihood median is only slightly better when using DPR. However, for the second scenario where the system starts with bad configurations, the QoS value is much better with the reconfiguration system. Hence, the DPR system is better at tracking a target with an a priori unknown trajectory. We also notice that the minimum likelihood measurement is higher with reconfiguration. This implies that the DPR system has less risk of losing the target than the one without DPR. All three scenarios show the same minimum and median values of the likelihood. This shows that our system is robust against its initial condition.

Figure 3.6a shows the third scenario case of a tracking using DPR. The green curve

Table 3.1 – Results of tracking

Tracking Scenario[1]	**Sub-trajectory**[2]	**Likelihood**	***DPR***	***No DPR***
Scenario 1	**Part 1**	***Min***	0.4156	0.4232
		Max	0.9970	0.9984
		Median	0.8074	0.8077
	Part 2	***Min***	0.3898	0.3889
		Max	0.9936	0.9938
		Median	0.6915	0.6910
	Part 3	***Min***	0.2580	0.2529
		Max	0.9835	0.9777
		Median	0.6103	0.6100
	Part 4	***Min***	0.2144	0.2128
		Max	0.9939	0.9812
		Median	0.5848	0.5790
Scenario 2	**Part 1**	***Min***	0.4301	0.1640
		Max	0.9939	0.9796
		Median	0.8025	0.4567
	Part 2	***Min***	0.3915	0.1198
		Max	0.9853	0.8620
		Median	0.6968	0.3708
	Part 3	***Min***	0.2582	0.1177
		Max	0.9934	0.9445
		Median	0.6119	0.2874
	Part 4	***Min***	0.2144	0.0868
		Max	0.9894	0.9872
		Median	0.5845	0.2954
Scenario 3	**Part 1**	***Min***	0.4156	0.4227
		Max	0.9976	0.9982
		Median	0.8063	0.8077
	Part 2	***Min***	0.3898	0.3897
		Max	0.9936	0.9966
		Median	0.6915	0.6922
	Part 3	***Min***	0.2580	0.2529
		Max	0.9835	0.9955
		Median	0.6103	0.6098
	Part 4	***Min***	0.2144	0.2114
		Max	0.9939	0.9748
		Median	0.5848	0.5803

[1]Scenarios are described in subsection 3.1.2
[2]Sub-trajectories are defined in 3.1.2

represents the real trajectory, while the red one shows the outputs of the reconfigurable Kalman system. In this example the Triggered-based reconfiguration (namely when the distance to the model exceed the threshold value) occurs once at $x = 6314$ when the trajectory radically changes. All other reconfigurations occur periodically at the end of each time-window and can provide the best KF that will be eventually selected according to the QoS comparisons, this is for instance what happens at $x = 3715$. Figure 3.6b shows the output trajectories for the three scenarios and without DPR. The differences between Figures 3.6a and 3.6b highlight the influence of reconfiguration on the Kalman filtering.

These improvements are allowed by the knowledge of the QoS value throughout the process. Thanks to this value, the system can choose the best model at a given time. Moreover, the likelihood also allows to know when the models are not adapted and the system can adapt early. This method requires good knowledge about the application, and extracting a QoS indicator may not always be straightforward.

3.1.3 Case study conclusion

This study first demonstrates the opportunity of using DPR to virtually extend available HW resource of an embedded system in the domain of Radar. The embedded system can dynamically implement tracking HW accelerators according to application requirements in order to speed-up the application execution while saving CPU time. The study also shows the importance of considering the QoS to drive the configuration. This point is rarely considered in the DPR literature since it requires transdisciplinarity. However, it is crucial to fully benefit from the expert knowledge (radar, signal processing in our case study). This knowledge can be captured and efficiently used only with a methodology that clearly separates concerns.

The next section presents another example of QoS-based dynamic partial reconfiguration which optimize another radar processing block.

3.2 A seamless DFT/FFT self-adaptive architecture for embedded radar applications

In radar tracking systems, the environment is composed of targets, clutter and other perturbations, which can be characterized by computing a range-Doppler map of the radar signal (i.e. a discrete version of the time-frequency analysis first described in [51]).

In practice, this function is composed of a matched filter and a series of Fourier transforms. This transform is the core of our study, it can be performed with different methods [52]. The main methods are the FFT [53], [54] and the classic DFT summation [55], [56]. The two approaches present different characteristics in terms of throughput, latency, processing and memory resources, that can benefit to different phases of application scenarios. In the context of multi-channel radars, the range-Doppler function becomes a major concern [57]. Indeed, although the AESA spatial degrees of freedom give extra information on the targets, this function must be computed for every channel, consuming a lot of logic resources. In this context, the DFT method used for the range-Doppler transform is fixed at design time. Hence, the system cannot benefit from different DFT architectures. DPR, which is proved to be particularly efficient in the signal processing field [58]–[60], permits to overcome this limitation.

This work presents a reconfigurable architecture which allows to use the proper version of DFT algorithms according to application needs, without the resource consumption overhead. This architecture limits the impact of the reconfiguration time by performing an original progressive reconfiguration that opens new perspectives. The second contribution is a method to take the reconfiguration decision at the right time in a radar application.

Section 3.2.1 details the DFT algorithm in the context of radar systems and explains the two main methods to compute it. Section 3.2.2 proposes a methodology to take efficient reconfiguration decisions at runtime. A methodology to create an adaptive reconfigurable DFT for radar, is described in section 3.2.3. Section 3.2.4 shows and comments the results obtained through a hardware in the loop (HIL) implementation on a case study. Section 3.2.5 concludes the work and draws directions for future work.

3.2.1 DFT options for radar processing

DFT algorithm

The DFT algorithm (defined by Eq. (3.10)) is used to extract the spectral distribution of a discrete signal.

$$A_k = \sum_{n=0}^{N-1} W_N^{kn} a_n \quad \text{where} \quad W_N^{kn} = e^{-i\frac{2\pi}{N}\times kn} \tag{3.10}$$

This equation allows the DFT computation of any size N of discrete signal. However, all the temporal samples a_n are required to compute a single spectral sample A_k, $k \in$

$[0, N[$. This algorithm is well known and many implementations exist in the literature, with complexities going from $\mathcal{O}(Nlog(N))$ for the FFT to $\mathcal{O}(N^2)$ for a direct implementation of Eq. (3.10). However, in some applications, this direct DFT implementation can be useful to order certain operations.

DFT concerns in radar processing

The DFT in radar processing is used to extract a Doppler frequency from the temporal signal. To this end, the system downsamples the input data and performs a DFT over the different sub-signals. The combination of a matched filter and this transform is known as 'range-Doppler processing'. Fig. 3.7-a) depicts this downsampling and the DFT processing to illustrate our statements. This manipulation is known as a 'corner turn' in radar and sonar domains [53], [54].

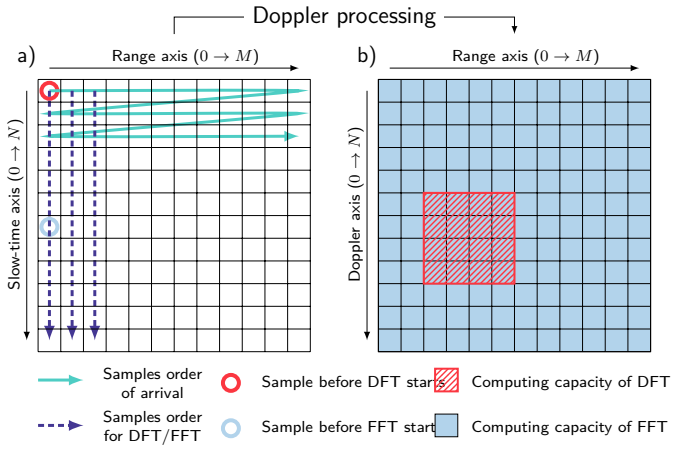


Figure 3.7 – Constitution of a range-Doppler radar map. a) down-sampling and DFT. b) Representation of the computing capacities of DFT and FFT.

Fig. 3.7 a) shows that to complete the first DFT, we need to wait for $(N-1) \times M + 1$ points to arrive. However, some DFT algorithms allow to begin the computation earlier. If we can process the input data as soon as we receive it, it is possible to reduce the DFT latency. Otherwise, we have to wait for a significant amount of data to arrive, which leads to a higher latency. This latency depends on the algorithm used for DFT.

Discrete Fourier transform for dataflow

An important property of the direct implementation of DFT, which we refer to as DFT in the rest of this study, is its ability to process inputs equally in any order. It is widely

used in radar systems to overcome the difficulty of processing the data in a different order from arrival (Fig. 3.7-a)).

Consuming the data in the natural order of arrival minimizes latency since computations start earlier and the $\mathcal{O}(N^2)$ computations are then covered as the data comes in. However, all partial sums of Eq.(3.10) need to be stored locally in the FPGA for every output, in order to keep the short latency benefit. But in contrast with FFT, DFT can compute a selected part of the spectral domain from the full time domain, reducing the number of computation while ensuring spectral accuracy. Hence, DFT allows to adapt the size of the explored spectrum to available storage and logic resources on FPGA. Eventually, we can summarize as follows:

- **pros:** dataflow processing, low latency, no need to buffer the input data, possibility to compute a selected interval
- **cons:** high DSP and BRAM consumption per output sample, limited range-Doppler exploration because of the limited output samples

Fast Fourier transform for block processing

In contrast to DFT, FFT algorithms use more efficiently both processing and memory resources, by means of computation results reuse. However, a drawback of FFT is the inability to process the inputs in order of arrival. With the DIF FFT algorithm for instance, $\frac{N}{2}+1$ input samples are required before the first computation can start. The main issue is the latency penalty. A second one is the incapacity to store all the input samples in the FPGA ($\frac{N}{2} \times number\ of\ distance\ gates \times number\ of\ channels$). Hence, an external DDR memory is used and introduces an additional latency, but it also frees up resources that can be used to compute the full spectral domain. Eventually, we can summarize as follows:

- **pros:** low resources consumption per output sample, full size range-Doppler exploration
- **cons:** high latency, data buffering is mandatory

Conclusion on DFT algorithms for radar applications

At radar startup, the system does not know the exact position of the tracked objects in Doppler and space domains. At this time of the mission, it is critical to have information on the full space to emphasize detection. Moreover, the latency is not critical since no tracking

is launched, thus a FFT is the right solution. Once detection is successful the system has to start tracking, which requires other algorithms (e.g. angle of arrival estimation, filtering with Kalman filters). The latency constraint is strengthened. Furthermore, the system now has a precise estimation of the target range and speed. So, the extensive overview on the range-Doppler domain can be exchanged for reduced latency, allowing the system to perform additional required functions. This trade-off on the exploration space of the two algorithms is illustrated in Fig. 3.7-b).

This describes the original idea of our study, but we still need to define the reconfiguration decision-making.

3.2.2 QoS aware reconfiguration controller

Upon startup, the system uses the FFT algorithm. Once detection is successful, we could change for the DFT algorithm. However, a successful detection cannot guarantee a positive detection in the next radar coherent processing interval (CPI). Indeed, a poor signal to noise ratio (SNR) or signal to clutter ratio (SCR) may result in the loss of the target. Such changes in SNR and SCR can come from radar cross section (RCS) changes, fluctuation loss or environment configuration.

The decision to reconfigure the system requires a more accurate QoS indicator than the simple Boolean one. We need a value which reflects the probability to successfully detect the target. A good solution is to use the probability of detection (P_d), which is in practice unknown but can be approximated from a quantifiable variable. One possible approach is to record the positive and negative detections of the target, and compute the target detection frequency. To use this approach, we need to ensure that the plots (i.e. positions where detection is positive) come from the same object. This can be achieved by a probabilistic data association filter (PDAF)[10]. This filter uses the log-likelihood of the innovation (ℓ) computed by a Kalman filter to determine if the plot is likely to be related to the observed target. The PDAF associates the plots to a track with a validate function $V(k)$ described in Eq. (3.11).

$$V(k) = \begin{cases} 1, \text{if } \ell(k) \geqslant \gamma \\ 0, \text{if } \ell(k) < \gamma \end{cases} \qquad (3.11) \qquad\qquad L_q(k+q) = \frac{1}{q}\sum_{i=1}^{q} V(k+i) \qquad (3.12)$$

Where : $\ell(k)$: log-likelihood of the innovation at plot k
γ : validation threshold

The detection ratio over q samples is given by Eq. (3.12). The target detection is confirmed when the frequency exceeds a defined threshold λ, so we can simply define the QoS criterion as the detection ratio: $f_{QoS}(k+q) = L_q(k+q)$. A PDAF is required in any radar tracking system, so the QoS function does not introduce a computational overhead.

The required configuration s_k can take one of the two states of $S : \{S_{DFT}, S_{FFT}\}$, which are the DFT and the FFT respectively, depending on the QoS criterion. Eq. (3.13, 3.14) describe the state of the configuration with regard to the QoS criterion. Fig. 3.8 gives another representation of this system through a cyclic graph and a transition hysteresis.

$$s_k = \begin{cases} S_{DFT}, & \text{if } S_{DFT} \cdot \bar{a} + S_{FFT} \cdot b \\ S_{FFT}, & \text{if } S_{DFT} \cdot a + S_{FFT} \cdot \bar{b} \end{cases} \tag{3.13}$$

Where: $$a = \bar{b} = f_{QoS}(k+q) < \lambda \tag{3.14}$$

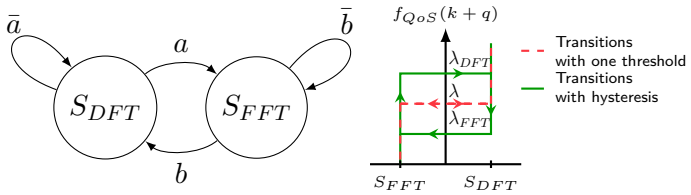


Figure 3.8 – Control graph and associated transition hysteresis

A transition based on Eq. (3.14) induces a risk of constant reconfiguration. For example, with $\lambda = \frac{2.5}{3}$, if the QoS value follows the sequence: $[\frac{3}{5}, \frac{2}{5}, \frac{3}{5}, \frac{2}{5}]$, the reconfiguration happens at every radar CPI. Therefore, we adopt an hysteresis model with two different thresholds λ for the two state changes. λ_{FFT} is used for the transition from DFT to FFT and λ_{DFT} for the reconfiguration from FFT to DFT. With $\lambda_{DFT} > \lambda_{FFT}$, the transition functions are given by Eq. (3.15).

$$a = f_{QoS}(k+q) \leqslant \lambda_{FFT} \; ; \; b = f_{QoS}(k+q) \geqslant \lambda_{DFT} \tag{3.15}$$

Fig. 3.8 shows that Eq. (3.15) results in a more stable controller, which is unlikely to cause ceaseless reconfigurations. This automaton is synchronous with the radar CPI end.

It is worth mentioning that the QoS criterion is specific to one target and must be computed independently for every distinguished target. Once all the criteria are computed, an external operator can choose where to place the DFT window at reconfiguration time. In this study, we suppose this operator wants the window to be centered on the last detected plot.

Since the concept and the benefits of the DFT reconfiguration have been exposed, we present an architecture allowing to efficiently implement a reconfigurable DFT.

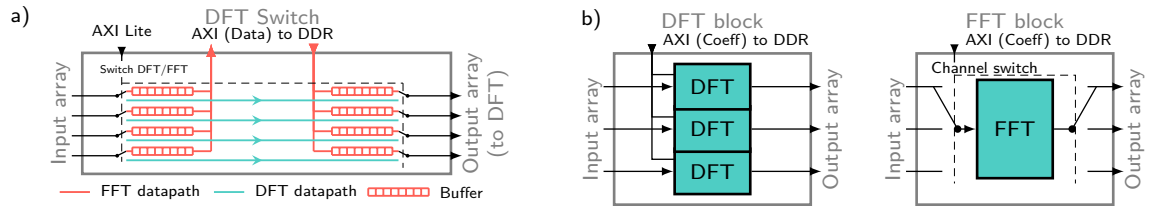


Figure 3.9 – a) Switch block to adapt input data ordering for DFT. b) The two configurations of the DFT IP.

3.2.3 Implementation of a reconfigurable DFT

Problem of channel reconfiguration

In a radar system, the data is processed from the sensors to the software in a dataflow way. The reconfiguration of part of this processing flow requires to interrupt the flow. The DPR introduces a delay which is not compliant with most of radar systems that may lose the track of targets during the interruption. However, we consider multi-channel radar, whose channels can be processed independently when DFT is performed. Hence, it provides an opportunity to reconfigure channels by blocks, from one single channel, up to all channels. Nevertheless, the number of channels to reconfigure at the same time determines the number of resources to reconfigure. Reconfiguration time increases linearly with the reconfigurable partition (RP) size. Our reconfiguration paradigm can be formalized as the following problem. *Determine the best trade-off between stopping the process for a long time to reconfigure all the channels and reconfiguring only one channel at a time, but running in a degraded mode with the other channels.* Fig. 3.10 shows a sixteen sensors radar system which implements DFT reconfiguration, where each RP performs four DFT

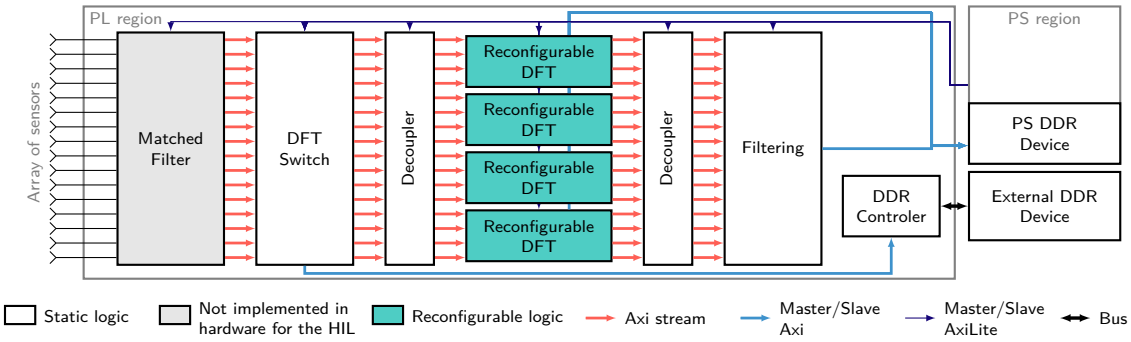


Figure 3.10 – Architecture used for the final results

to divide the reconfiguration time in four steps. During this process, the radar system is still running with 3/4 of the channels.

Data reordering

As introduced in Sec. 3.2.1, DFT and FFT inputs are not processed in the same order. For the DFT, samples are processed in arrival order without having to reorder input data. Unlike DFT, FFT receives the data after downsampling, and thus store batches of data in DDR until it has enough data in the processing core. It means that a switch is required before the reconfigurable DFT to act either as a passthrough or as a store and load device. Fig. 3.9-a) represents this block which uses a small amount of resources, thus we keep it out of the RP. However, this block is mandatory when using FFT, to deal with the 'corner turn', but AXI (Data) is used only with the FFT configuration. An additional AXI lite slave port is required to configure the switch block in FFT or DFT modes.

Architectures of DFT/FFT configurations

An efficient DPR requires that the configurations sharing the same RP have similar amount of resources since the RP is sized for the worst case, so a mismatch may result in an excessive waste of hardware resources.

The DFT uses a lot of memory but only few DSP per channel whereas an efficient FFT implementation demands a lot of memory but also a lot of DSP. So we can balance the resource consumption disparity by implementing n DFT per FFT block. This implies no latency issue since most of this latency is due to data movements with the FFT model. Fig. 3.9-b) shows how the configurations design can balance resources usage by using the same FFT core for $n = 3$ channels.

It is important to note that if only the FFT needs to access to the DDR for the Data

though an AXI (Data) bus, the situation is different for the DFT since only the DFT needs to read the coefficients from the DDR using an AXI (Coeff) bus which is unused with the FFT configuration. Merging the AXI (Coeff) and AXI (Data) requires more Mux resources and architecture complexity, hence we keep a solution with two distinct bus.

A second consideration is the DDR bandwidth limit. Indeed, the pipelined versions of the FFT can read a new input at each clock cycle. If we can read up to N values per clock cycle from the DDR, it is useless to implement more than N FFT.

Conclusion on the DFT implementation

Multiple parameters impact the implementation results (e.g., DFT window size, computations parallelism, reconfiguration speed and channel number per FFT). Therefore, the design space exploration is made by means of a HLS specification that allows to get resource / performance estimations in a reasonable amount of time. In this study, we implement a versatile IP resulting in different implementation from preprocessing directives. We have three distinct configurable specifications: a DFT, a wrapper to the FFT of Xilinx with the pipeline configuration, and a homemade pipelined FFT with resources consumption and latency close to the proprietary IP. We implement our own pipelined FFT on Vivado HLS to generate a HLS synthesis estimation of resources consumption. Indeed, the Vivado FFT uses incorrect precomputed HLS results with the tool current version (2019.1) as indicated in Table 3.2 as the *HLS estimation quality*. Our HLS code allows to choose the number of channels per FFT in the reconfigurable IP.

3.2.4 Case study

System description

To study the performance of our architecture, we implement the system described in Fig. 3.10 on a ZCU102 board. This board includes a Zynq Ultrascale+ circuit, two DPR are attached to the ARM (PS) processor and FPGA (PL) respectively.

The input signal is a synthetic radar signal, transporting information on a target through delay, Doppler frequency and direction (phase shift). This signal is injected at the DFT Switch input of Fig. 3.10. The filtering is a simple beamformer which projects the signal in a spatial direction, and write the result to the PS DDR memory. The detection is then performed with a CA-CFAR [83] (Cell-Averaging Constant False Alarm

Rate) implemented in software. The reconfiguration controller defined in Section 3.2.2 is implemented in software as well.

We use the Gazebo simulator[84] to compute the target movement and provide a graphical output for the HIL demonstrator. The simulator sends the target position and speed to a Python 3 process that generates the signals. This process sends the signal through an Ethernet link to the board. The reconfigurable modules of the FPGA computes the DFT and the static part filters the data with the beamforming. Detection is executed by the Zynq ARM as well as the reconfiguration controller (described in Sec. 3.2.2) which is updated accordingly. The board sends the output signal and the detection plots to our Python 3 interface through Ethernet, for graphical plots.

PCAP API improvement

DPR is performed through the processor configuration port (PCAP). But we have modified the Xilinx PCAP API to add a non-blocking reconfiguration command. When the reconfiguration is launched, the task is suspended. When the PCAP is done, a CSUDMA interruption resumes the task. Furthermore, this API greatly reduces the reconfiguration time by avoiding data cache flush. Unlike the Xilinx API, we control the memory range and perform the flush operation out of the reconfiguration time. The impact on performance is given in the Sec. 3.2.4: Performance results.

Scenario description

We choose to highlight our architecture benefits in terms of performance and adaptability. Without loss of generality and to avoid interference with phenomenons out of the scope of this study, the scenario features one target in white noise, without clutter. This simplification does not invalidate the key concept since the DFT reconfiguration process would be the same with a target in a clutter. Besides, changing the architecture to a multi-target version would only add a selection method which is application specific and usually implemented out of the radar system itself. However, the simplification results in a simpler QoS function. The association result is considered as correct when a plot is detected. We fix the QoS thresholds $\lambda_{DFT} = \frac{3}{5}$ and $\lambda_{FFT} = \frac{1}{5}$. This configuration implies that the target is still present in the DFT scope (and will likely be detected again) if at least three detections were successfully performed in the last five radar CPI. However, if less than two detections are successful, the system should explore the entire range-speed space. To observe the reconfiguration stability, we use the Swerling I model of RCS [85].

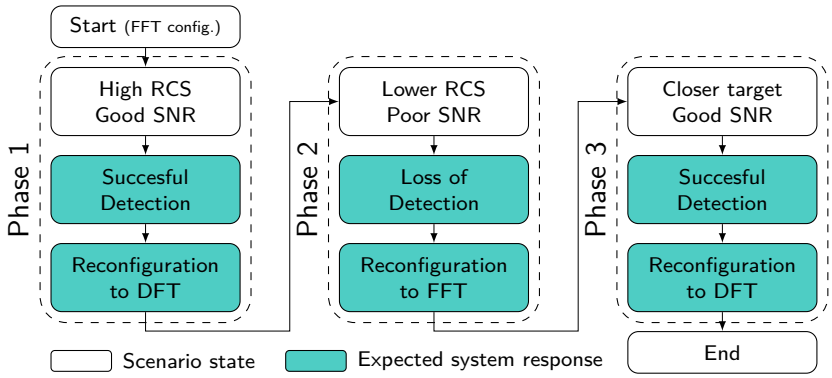


Figure 3.11 – Case study scenario

The scenario is composed of three phases depicted in Fig. 3.11. At the beginning of the mission, the target is distant. The radar system does not know the target position in the range-Doppler space and is in FFT configuration. At first, the RCS ensures a good SNR. At this time of the mission, our system should detect the target and switch to DFT configuration. Later, the target orientation changes, lowering the RCS. The SNR falls, leading to the target loss. The system should then switch to FFT configuration. The target speed direction changes in the time it was lost. When the target is close to the radar, the system detects it again (at a different place in the range-Doppler map), it should switch to DFT configuration and the mission ends. We use this scenario to test our HIL system and provide a demonstration video available online[86].

Implementation results

As we stated in Sec. 3.2.3, the whole architecture is generated with HLS codes. The main parts are the DFT Switch and the different configuration of DFT. The implementation can be parameterized with six variables: (1) N_s is the maximum signal size (radar CPI), (2) N_d is maximum DFT size, (3) N_c is total number of channels, (4) N_{dfts} is number of samples per DFT, (5) N_{ipc} is channels per IP, (6) N_{fft} is FFT per IP (FFT mode). The first three parameters depend on the application. We choose the parameters as follow in our case study:

(1) $N_s = 16000$ (2) $N_d = 1024$ (3) $N_c = 16$

The next parameters need exploration to choose the best channels slicing and to control its impact on the radar performances. In a first step, we carry out different Vivado HLS

Table 3.2 – Resources consumption of the different configurations

IP configuration for four channels	Precision	LUT	FF	BRAM	DSP
DFT (32 × 32 samples)	full	4889	2648	43	48
Xilinx FFT (8 to 1024 samples)	scaled 24bits	6697	9136	6	24
FFT (8 to 1024 samples)	scaled 24bits	7496	7407	14	24

Previous HLS estimation quality: Accurate | Approx. | False

synthesis to observe the number of DFT we can implement for one FFT to have equivalent resources usage. The DFT and FFT sizes result in an optimal ratio of four DFT per FFT (resource consumption from logical synthesis in Table 3.2). Furthermore, the maximum DDR bandwidth allows us to read up to 4 values per clock cycle. Therefore, we implement one FFT per IP, for a total of four IP. We determine the best configuration:

(4) $N_{dfts} = 1024$ (5) $N_{ipc} = 4$ (6) $N_{fft} = 1$

In our study, the PRF is not a relevant parameter and remains unchanged. The two important observed metrics are the latency and the size of the speed-range domain.

The full architecture is depicted in Fig. 3.10. The input signal is injected to the DFT Switch and the matched filter is performed beforehand by a Python 3 code. This architecture is based on the parts described in Sec. 3.2.3. Decouplers are connected to the RP to guarantee isolation at reconfiguration time. These decouplers can isolate the four partitions independently to allow the reconfiguration of only a subset of the channels.

Performance results

With this configuration, the observed system response meets the expected response described in Sec. 3.2.4. We measured a latency of $6.6ms$ for the DFT configuration, which coincides with the sum of the CPI and the beamformer latencies, so DFT latency is negligible compared to the other latency sources. However, with the FFT configuration, the latency is higher with a total of $9.3ms$. The major improvement offered by this architecture is the $2.7ms$ measured gain on the system latency when using DFT instead of FFT, $\frac{2.7}{6.4} \times 100 = 42\%$ of the CPI in our example. It means that the system is more responsive and this time is available for complex algorithms for tracking.

The RP reconfiguration time of each reconfigurable DFT block takes from $4.4ms$ to $5.8ms$ with the Xilinx PCAP API, and $2.2ms$ to $2.8ms$ with our modified API. This reconfiguration time is smaller than the CPI of $6.4ms$, enabling to reconfigure without

interruption of the radar processing flow. Indeed, although an expected degradation of the signal quality is observed during the four cycles needed to reconfigure all the RP, the radar signal consistency is preserved.

3.2.5 Case study conclusion

To the best of our knowledge this work presents the first self-adaptive architecture in the embedded radar field that tracks the best trade-off between latency and detection space by means of smooth DFT / FFT reconfiguration per channels group. Our reconfigurable architecture allows to take full benefit of the conventional DFT and FFT respective strengths. The decision method is the first contribution detailed in Sec.3.2.2, it relies on the use of QoS specific to our Radar application. The second contribution is a dynamically reconfigurable architecture which is optimized for the implementation of DFT / FFT configurations as well as fast and flexible enough to avoid interruption of the radar function during transitions.

Through a case study implemented by means of a realistic HIL approach, we demonstrate the efficiency of the architecture implementation. Our solution is valid for a large range of radar applications and can be parameterized to fit with different system configurations (e.g. number of channels, ranges, etc.). In real-life applications, it is possible to observe objects at different speeds and ranges. So a future work will allow to isolate and process several areas in the range-speed map. The proposed approach remains relevant, but modifications have to be considered in the DFT IP and in the downstream processing.

3.3 Conclusion

Through two cases studies, we showed the improvements that can be made by reconfigure computing hardware based on QoS criteria, as opposed to external commands. The reconfigured algorithms and QoS criteria were specified by application experts using knowledge that is unknown to the hardware experts. However, the application expert does not have the knowledge required to efficiently implement reconfiguration. Thus, this work also showed that good communication between hardware and application designers is essential to achieve maximum system performance. In this chapter, we worked on processing for which there are several versions with different quality results, but this is not always the case. In the next chapter, we provide an example of modifying an algorithm

to make it reconfigurable.

Chapter 4

Algorithm adaptation to benefit from reconfiguration

STAPLE: a Space-Time Adaptive Processing Learning-based Enhancement

Contents

In this chapter, we present a modification of an existing algorithm to make it reconfigurable and to be able to envision reconfigurable architectures to optimize it. This algorithm is the space-time adaptive processing (STAP), which is proved to be an efficient filter for clutter and jamming rejection. However, the dimensions of the whitening matrix used to filter out interferences can be very large to ensure good disturbance rejection capabilities. These matrix dimensions result in a huge computing complexity, a possibly ill-conditioned matrix and a high number of required training samples. Previous work propose to cope with this complexity by projecting the signal in a space with reduced dimensions, which decreases computation cost while ensuring a correct quality of processing. However, these projections require eigendecompositions which are also computing intensive and subjects to computational errors. We introduce a new alternative and complementary method which uses a reduced version of the sample matrix in order to choose the best STAP dimensions at run-time by means of supervised learning. In this work, we present a method based on a convolutional neural network and a random forest to reduce the problem of the STAP complexity. We demonstrate the effectiveness of this concept on synthetic noises and interferences with a reduction of more than eight times the computing load of the STAP at the cost of a small loss of processing quality.

When detecting targets, radar systems encounter a range of perturbations that must be filtered out to preserve detection capabilities. The power of these interferences (e.g. clutter, jamming) can be times stronger than that of the signal of interest. The properties of these perturbations make 1-D filtering (e.g. MTI) insufficient. Modern sensors are composed of many antennas (flat or conformal phased arrays), allowing the use of Space-Time Adaptive Processing (STAP)[61]. STAP discriminates the interferences based on correlations in the space-Doppler domain. While this two-dimensional filter offers considerable signal noise clutter ratio improvements, it comes at the price of a huge computational complexity and memory usage. In fact, the 'full' STAP algorithms include the estimation of a NM matrix and its inversion for every resolution cell under inspection and Doppler gate, with N the number of antenna elements and M the number of backscattering pulses. The practitioner encounters several drawbacks when trying to apply STAP in real cases.

- First, obviously the memory needed to store the STAP code and the data is a limiting parameter in particular for UAV-borne sensors for which the footprint of the signal processing embedded subsystem is limited.
- Similarly, the time needed to process the data for a full STAP processing could imply that the processing component does not have enough time to inspect the entire

scene or/and the Doppler space. Moreover the computational load could induce that the radar is only devoted to the target detection while modern radar systems are designed to perform several missions, such as target tracking or scene imaging through Synthetic Aperture Radar (SAR) processing for instance.

- As seen below, the covariance matrix estimation involves a large number of ancillary data (at least equal to matrix dimension; optimally twice) with the usual assumption that time-Doppler statistical properties of the interference are the same as for the tested cell range-Doppler gate and do not contain any spurious target, hypotheses that could not be fulfilled when the number of ancillary data is large.
- Finally, the matrix inversion is an overlooked problem since the inversion of such large random matrices is subject to significant computing errors, due to a massive accumulation of numerical quantification noise and round off. Indeed, the ratio between the maximal and minimal eigenvalues is more likely to be large, hence increasing the matrices conditioning number as seen below. This implies that even if the estimated covariance matrix is close to the true matrix, thus the inverse of the estimated matrix is far from the true one and the radar data whitening is not efficiently performed.

In practice, experts use reduced STAP algorithms (see [62]–[66] among an important literature), trading quality of results for computing efficiency and numerical stability. In some cases, reduced N and M dimensions are fixed by the user at design time. In other cases, an adaptive reduction is performed. However the new matrix dimension is based on the estimated random eigenvalues subject to computational round off and errors (see [67] for a discussion on random eigenvalues of large covariance matrices). Thus for operational purposes, it is of primary importance to design an efficient filter. Ideally, dimensions should be adapted depending on the interferences correlations in the space-time domain (not necessary on the eigenvalues). The different interferences, such as jammer and ground clutter, exhibit different properties; hence they have various space-time correlations. Since these correlations are *a priori* unknown, the usual solution is to set $N = M$. But we can further reduce the dimensions of the STAP filter by predicting the most appropriate N and M values at runtime. For instance, a broadband jammer can be effectively filtered out using only the spatial dimension ($N > 1$, $M = 1$). On the other hand, the clutter ridge is located along the line $M = N$ (e.g. [61], [87], [88]). In this work, we assume that, if we can first observe such patterns in the covariance matrix, then a machine learning process could extract fine information to adjust the N and M dimensions while the system

is running. Machine learning has already proven to be very effective in radar processing [89], [90]. Thus, we propose STAPLE, a method based on machine learning prediction models to determine the most adapted N and M dimensions of the STAP before performing the costly computation. Our prediction models base the inferences on the condensed observation of the interference (being the input of our prediction models) offered by a reduced dimension covariance matrix, this matrix being first estimated in the online process (and then avoiding the tedious estimation of the full dimension matrix). The prediction models are based on a combination of light Convolutional Neural Networks (CNN) and Random Forests (RF), which have good characteristics to be implemented within highly parallel and pipelined architectures (e.g. FPGA or GPU implementations[79]–[81]). We compare our approach to the standard $N = M$ approach using the Improvement Factor (IF), matrix condition number and computing complexity as metrics. Several authors [68]–[70] proposed other approaches for the dynamic allocation of STAP dimensions without machine learning. The approach proposed in [68] requires performing STAP with the maximum number of cells of one dimension for both STAP dimensions and computing the ratio of obtained residual noise levels. The solution presented in [70] uses an iterative algorithm which performs a series of channels tests/selections to select the most relevant antenna elements and backscattering pulses among the N/M dimensions. Our solution holds a lower computing complexity and exhibits more computation parallelism, thus shorter response times.

STAPLE can be considered as Knowledge-Aided STAP (see [91]–[93] among recent literature), where the knowledge is assimilated offline during learning.

As previously stated, our study focuses on airborne radars, which are confronted with a constantly changing environment. These systems can benefit the most from the proposed approach as they are often limited in terms of hardware resources, memory and computing time. For the sake of generality, we consider a monochromatic radar waveform as well as a chirp waveform to achieve high range resolution radar.

Section 4.1 describes the estimation of the covariance matrix from a radar data cube and the impact of the size of the STAP filter on the computing complexity. Section 4.2 details the selection of our dataset and defines the physical models used to simulate clutter, jamming and Radio Frequency Interference (RFI). These models describe the covariance of interferences for a monochromatic waveform and a chirp waveform with range compression as stated above. Section 4.3 proposes to infer the optimal distribution of the filter in the spatial / Doppler domains using a reduced covariance matrix and describes the prediction

models selected for the tests. Section 4.4 presents the predictions accuracy and highlights the gains obtained on the computing complexity of the STAP filtering compared to the standard approach. Section 4.5 concludes this work, discusses the limits of the method and draws directions for future work.

4.1 STAP and Sample covariance matrix estimate

In this work, we consider the detection of a target through an air / UAV borne radar, using a monochromatic pulse or a chirp to achieve higher resolution cell. Then the isorange/isoelevation is a circle centered on the radar covering 360° for the azimuth angle (see Fig. 1.3b). Obviously this isorange is given the round trip travelling time of the emitted pulse (as depicted in Fig. 6 of [4] for instance). We do not make any assumption on the radar frequency band for the sake of generality. STAP processing involves filtering out electromagnetic perturbations for a given range (or elevation for an air/UAV borne radar) and a given azimuth. It deals with the data cube gathering the backscattered voltage for each range/elevation gate (fast time), the number of range gate being limited by the propagation conditions of the EM wave, for each pulse (slow time) the maximum number of pulses denoted M_f being fixed either by the operator or by the storage capability and obviously each antenna element. The maximum number of antennas, denoted N_f, is set by the sensor design (see Fig. 4.1). In what follows (N_f, M_f) refers to the full STAP dimensions and Q, defined as the product of the number of antennas and the number of pulses is referred as the degrees of freedom of the STAP filter and as seen below, it is a key parameter for operational STAP approaches (e.g $Q = N_f M_f$ in the full dimension case). The first step of STAP involves stacking for each range/elevation gate, the received voltage of each pulse of all the antenna elements into an array denoted $\mathbf{y}_r$ (the subscript is an integer r referring to the range gate). Figure 4.1 shows that the measurement vector $\mathbf{y}_r$ of size $N_f M_f$ is created from a slice of the radar data cube. The decision on the target presence is based on the filtered value of $\mathbf{y}_r$, that is on $\mathbf{w}^H.\mathbf{y}_r$, H denoting the Hermitian transpose. The STAP weights, denoted $\mathbf{w}$, are given (under the Gaussian assumption, see [4], [61]) by:

$$\mathbf{w} = \gamma \mathbf{R}_r^{-1}\mathbf{s} \quad \text{with} \quad \mathbf{R}_r = E\{\mathbf{y}_r \mathbf{y}_r^H\} \tag{4.1}$$

Where γ is a constant that does not affect the detection process and $\mathbf{s}$ is a $N_f.M_f$ space-time steering vector which performs a matched filtering in the space-Doppler domain and then depends on the target Doppler (i.e. target velocity) and azimuth. $\mathbf{R}_r$ is the covariance

matrix of noises and interferences ($E\{\}$ is the mathematical expectation). Then, $\mathbf{R}_r$ as its estimate, denoted $\tilde{\mathbf{R}}_r$, are Hermitian. In order to carefully define the statistical properties of $\mathbf{R}_r$ and their effect on the sample covariance matrix in Section 4.2.1, Eq. (4.2) gives the indexing of a matrix $\mathbf{R}_r$ of dimensions $(N_f M_f \times N_f M_f)$ with $N_f = M_f = 2$.

$$\mathbf{R}_r = \begin{bmatrix} r_{(0,0),(0,0)} & r_{(0,1),(0,0)} & r_{(1,0),(0,0)} & r_{(1,1),(0,0)} \\ r_{(0,0),(0,1)} & r_{(0,1),(0,1)} & r_{(1,0),(0,1)} & r_{(1,1),(0,1)} \\ r_{(0,0),(1,0)} & r_{(0,1),(1,0)} & r_{(1,0),(1,0)} & r_{(1,1),(1,0)} \\ r_{(0,0),(1,1)} & r_{(0,1),(1,1)} & r_{(1,0),(1,1)} & r_{(1,1),(1,1)} \end{bmatrix} \tag{4.2}$$

Target detection involves choosing the modelling of the measurements $\mathbf{y}_r$ as a function of covariance matrix between the two hypotheses of Eq. (4.3) through Neyman-Pearson test for instance.

$$\begin{aligned} \mathcal{H}_0 : \quad & \mathbf{y}_r = \mathbf{R}_r^{1/2}\mathbf{u} \\ \mathcal{H}_1 : \quad & \mathbf{y}_r = \mathbf{x}_r + \mathbf{R}_r^{1/2}\mathbf{u} \end{aligned} \tag{4.3}$$

Where $\mathbf{x}_r$ is the the signal of interest array, that is a possible target response, and $\mathbf{u}$ is a zero-mean random vector of normal distribution. Thus, $\mathcal{H}_0$ is the absence and $\mathcal{H}_1$ is the presence of a target. As previously stated, the STAP filtering implies to verify that the filter data can be modeled as a white complex noise vector (i.e. $\mathbf{u}$) in Eq. (4.1). However, in operational cases $\mathbf{R}_r$ is unknown. Hence, the sample matrix inversion technique of STAP filtering estimates an approximation of the exact covariance matrix from collected samples in order to have an estimate of the noise / clutter time / space statistical properties. In fact, to have an estimation $\tilde{\mathbf{R}}_r$ of the covariance matrix $\mathbf{R}_r$ at range r, we sum the covariance matrices obtained on several range gates as in Eq. (4.4).

$$\tilde{\mathbf{R}}_r = \frac{1}{|K|} \sum_{k \in K} \mathbf{y}_k \mathbf{y}_k^H \tag{4.4}$$

where K is a set of unique range gate indices (that does not contain r), with a total of $|K|$ training samples.

Since the matrix $\tilde{\mathbf{R}}_r$ is Hermitian, the upper triangle contains the whole information on the signal. To filter a sample at range r, we usually use several neighbour range covariance matrix. For example in Fig. 4.1, the measurements $\mathbf{y}_5$ are filtered using covariance matrices computed from measurements $\mathbf{y}_2$, $\mathbf{y}_3$, $\mathbf{y}_7$ and $\mathbf{y}_8$. [94] proved that the total number of training samples must be greater or equal to $2Q$ for a proper training of the STAP filter, which demonstrates the interest in N and M reduction as stated in introduction.

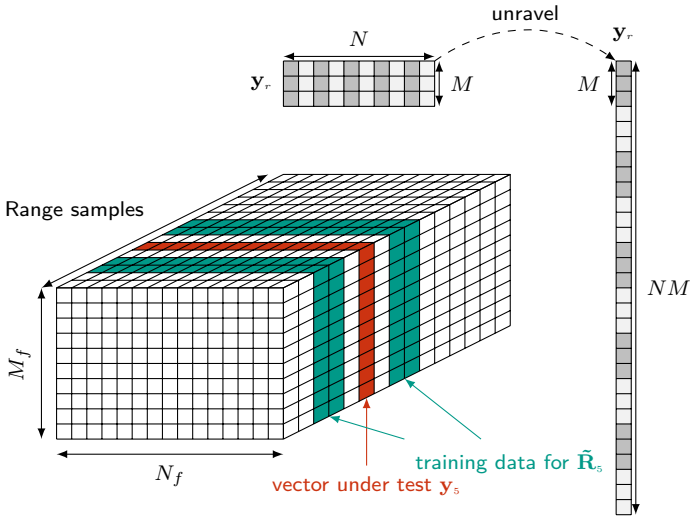


Figure 4.1 – STAP data cube and creation of $\mathbf{y}_r$

The obtained matrix carries information on the noise contained in the radar signal. This matrix also sets the complexity of the STAP processing as the costliest operation consists in inverting this matrix. The inversion of this matrix holds a $\mathcal{O}(Q^3)$ complexity. It is exponentially complex to compute the inverse of this matrix as we increase one of the dimensions N and M. [68] emphasized that the value of Q is constrained by three values: Q_1 imposed by the computational load, Q_2 imposed by the limited number of cells available for training and Q_3 imposed by the number of antenna elements and slow-time cells available for processing, thus we have to verify $Q \leq \min(Q_1, Q_2, Q_3)$. To respect these constraints on Q, we must reduce the sample vector $\mathbf{y}_r$, the steering vector $\mathbf{s}$ and the covariance matrix $\mathbf{R}_r$ sizes. Equation (4.5) projects the problem in a new space with a matrix $\mathbf{T}$ of dimensions $(N_f M_f \times NM)$.

$$\mathbf{y}'_r = \mathbf{T}^H \mathbf{y}_r, \ \mathbf{s}' = \mathbf{T}^H \mathbf{s}, \ \mathbf{R}'_r = \mathbf{T}^H \mathbf{R}_r \mathbf{T} \tag{4.5}$$

Where $\mathbf{y}'_r$, $\mathbf{s}'$ and $\mathbf{R}'_r$ are the projections of the sample vector, the steering vector and the covariance matrix, respectively. For the sake of simplicity, $\mathbf{T}$ will be omitted in the remainder of this work and is implicit when $NM < N_f M_f$. Different techniques can be used to create the projection matrix. Some methods use an eigendecomposition of the matrix $\tilde{\mathbf{R}}_r$ but this is time consuming and prone to errors with a large condition number. Another solution is to recombine antennas and slow-time samples into new channels (see [95] for an example of dimension reduction by antennas recombination). Even though this

method preserves a theoretical filtering improvement close to the full STAP one, it also significantly decrease the stability of the inversion process of matrix $\mathbf{R}'_r$. The solution used in our method is to select the first N and M channels with $\mathbf{T}$. Equation (4.6) shows how the entries of matrices $\mathbf{T}$ of size $(N_f M_f \times NM)$ are created in this work.

$$t_{(n,m),(n1,m1)} = \begin{cases} 1 & \begin{array}{l} n = n1 \\ n < N \\ m = m1 \\ m < M \end{array} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

To evaluate the performance of our proposed approach, the profit and loss of STAPLE, we consider three points of view and derive three different metrics.

1. The first is the efficiency of the filtering, i.e the capability of the STAP filter to reject out the disturbances. For measuring the filtering performance, we use the improvement factor defined as the ratio between the input and the output Signal-to-Interference-plus-Noise Ratio (SINR), which is computed with Eq. (4.7).

$$IF = \frac{SINR_{in}}{SINR_{out}} = \frac{\mathbf{w}^H \mathbf{s}^H \mathbf{s}\mathbf{w} \cdot \mathbf{tr}(\mathbf{R}_r)}{\mathbf{w}^H \mathbf{R}_r \mathbf{w} \mathbf{s}^H \mathbf{s}} = \frac{\mathbf{s}^H \mathbf{R}_r^{-1} \mathbf{s} \cdot \mathbf{tr}(\mathbf{R}_r)}{\mathbf{s}^H \mathbf{s}} \tag{4.7}$$

   Obviously, this quantity has to be higher than 1 (or 0 in dB). The improvement factor can be defined for the full dimension but also for a reduced version by replacing $\mathbf{s}$ and $\mathbf{R}_r$ by $\mathbf{s}'$ and $\mathbf{R}'_r$ in Eq. (4.7). The IF is expected to be lower in the reduced dimension case and for this reason we define:

$$G_{\mathcal{IF}} = \frac{IF_r}{IF_f} \tag{4.8}$$

   where IF_f and IF_r are the improvement factors of the full and reduced STAP respectively. $G_{\mathcal{IF}}$ is smaller than 1, but must be the closest possible to unity to mean that the reduced dimension do not induce a filtering capability loss.

2. The second point of view is the estimation of the inverse covariance matrix. The conditioning number κ of a matrix to be inverted is defined as Eq. (4.9).

$$\kappa = \frac{|\lambda_{max}|}{|\lambda_{min}|} \tag{4.9}$$

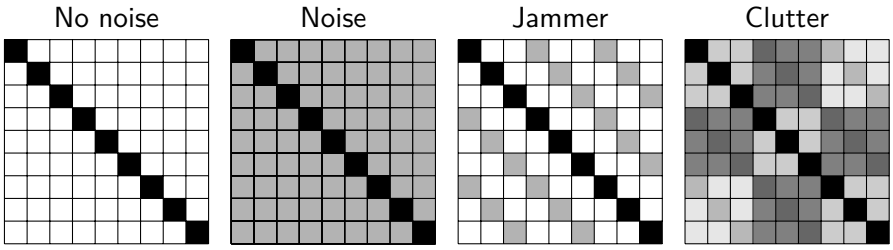


Figure 4.2 – Covariance pattern examples

Where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of the covariance matrix (R_r or R'_r), respectively. κ is also higher than 1 and must be close to unity to ensure a stable inversion, that is an inverse of the estimated matrix close to the inverse of the exact matrix. For reduced dimension, the conditioning number is higher than in the full dimension case (since we work on subspaces). We define as metric:

$$G_{\mathcal{CN}} = \frac{\kappa_r}{\kappa_f} \tag{4.10}$$

where κ_f and κ_r are the conditioning number of the full dimension and the reduced dimension, respectively. For a higher stability this ratio must be the closest to 0.

3. The last metric is a computational point of view. The overall complexity of our decision algorithm should be less than the saved complexity on the original STAP computation. Equation (4.11) gives the gain on the complexity $G_{\mathcal{O}}$ compared to the full STAP that can be achieved by a prediction system.

$$G_{\mathcal{O}} = \frac{N^3M^3 + X}{N_f^3M_f^3} \tag{4.11}$$

Where X is the complexity of inference with the decision method, i.e. during the target detection (the high offline computational complexity, i.e. the learning step, has not to be account in this metric), and N and M are the selected reduced STAP dimensions. The learning of such process requires a dataset containing different interferences and the associated NM optimal dimensions.

Table 4.1 – Notation summary

v_a/h_a	Aircraft velocity/altitude.
δ_r	Range cell resolution
$\vec{k_p}$, k_p, λ	EM wavevector/norm/wavelength.
θ_J/ϕ_J	jammer elevation/azimuth.
θ_{RFI}/ϕ_{RFI}	RFI elevation/azimuth.
τ/T_r	Pulse duration/pulse repetition period
ω_c/K_ω	carrier angular frequency/compression factor.
B/c	Bandwidth (chirp)/speed of light in the air.
$\lvert A(\phi)\rvert^2$	Mean far field radiation intensity per unit of time time.

4.2 Covariance matrix dataset

From the aforementioned observations on the covariance matrix, we propose the following method: The first step is the creation of the upper triangle of the reduced-size covariance matrix $\tilde{\mathbf{R}}_{r,p}$ (with dimensions N_I and M_I the values of these hyperparameters being discussed in 4.3.2). Next, a CNN extracts information from patterns in this matrix. Then, this CNN outputs the optimal dimensions N and M. Eventually, the system uses the predicted dimensions NM to create a covariance matrix and run STAP. Supervised learning requires a dataset of many samples to train and test learning models. Our system relies on the prediction of optimal filtering dimensions for each specific electromagnetic interference. Hence, Section 4.2.1 defines several interference models and the effects of their physical properties on the covariance matrix $\mathbf{R}_r$. Then, the rules for optimal dimensions selection are described in Section 4.2.2. Eventually, Section 4.2.3 details how our learning set is build up.

4.2.1 Physical models

In what follows, we consider a plane or an UAV flying at h_a with velocity v_a. As stated in introduction, we first consider a monochromatic waveform, that is a radar emitting a pulse of duration τ only on a carrier angular frequency ω_c (usual assumption when dealing with STAP). Thus, the electromagnetic wave propagates as $e^{j(\omega_c t - \vec{k}p\vec{r})}$. In this case, the range cell resolution is $\delta_r = c\tau/2$, c being the speed of light. In order to achieve high

range resolution, the emitted wave form is a chirp, this chirp being centered at ω_c, the compression factor is equal to K_w and thus the instantaneous angular frequency is given by $\omega = \omega_c + K_\omega.t$ for $-\tau/2 \leq t \leq \tau/2$. The bandwidth is given by $B = K_\omega \tau$ and the resolution cell is $\delta_r = c/2B$. The antenna elements are assumed to be equally spaced by the half wavelength at ω_c, i.e. $d = \lambda/2$ with $\lambda = 2\pi.c/\omega_c$ (usual configuration). The Pulse Repetition Period is denoted T_r. We use the side-looking hypothesis and thus the azimuth reference, i.e. $\phi = 0$, is given by the normal to the linear antenna array. This assumption eases the selection of the optimal (N, M) couple in the dataset (cf. Section 4.2.3). The physical/radar parameters are summed up in Table 4.2.

Several noises and clutters must be computed to train and validate our intelligent system. Our simulation features one noise and three interference types:

- Complex gaussian noise
- Ground clutter
- RFI
- Wideband jammer

We describe the noises and interferences equations in the following Sections, with and without pulse compression.

Complex Gaussian noise

In radar systems, complex Gaussian noise (that is a complex random process with independent real and imaginary part, each following an identical and zero mean normal distribution) can come from different sources, such as thermal noise or natural atmospheric noise (see [96] for a description). Its covariance matrix can be represented by a diagonal matrix as shown in Eq. (4.12).

$$r_{(n,m),(n_1,m_1)} = \begin{cases} \sigma^2 & \text{if} \quad n = n_1 \text{ and } m = m_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

Where σ^2 is the complex Gaussian noise variance. i.e the noise power [97].

Ground clutter

In order to calculate, the clutter contribution for each antenna and each pulse, we make the usual assumption of clutter patch-to-patch independence (see [98]). Fig. 1.3b illustrates the clutter calculation geometry. For the monochromatic wave radar, we straightforwardly use the calculation of [98] i.e. the clutter contribution to the covariance matrix $\mathbf{R}_r$ is

computed from the series defined by Eq. (4.13) which quickly converge (we use this fast calculation due to the large amount of data to be generated during the learning, see Section 4.2.3).

$$\begin{aligned} r_{(n,m),(n_1,m_1)} = C_0 J_0(|z|) + \sum_{k=1}^{\infty} C_{2k} J_{2k}(|z|)\cos(\psi_{2k} + \varphi_{2k}) + \\ \sum_{k=0}^{\infty} C_{2k+1} J_{2k+1}(|z|)\sin(\psi_{2k+1} + \varphi_{2k+1}) \end{aligned} \tag{4.13}$$

Where $J_\nu()$ is the Bessel function of first kind of order ν and

$$\begin{array}{l|l} \beta = 4v_a.T_R/\lambda & \begin{Bmatrix} A_k \\ B_k \end{Bmatrix} = \tau^2 \int_{-\pi}^{\pi} |A(\phi)|^2 . \mathrm{sinc}^2(\Omega_C.\tau/2) \begin{Bmatrix} \cos(k\phi) \\ \sin(k\phi) \end{Bmatrix} d\phi \\ \xi = \pi\sqrt{1 - (\frac{h_a}{r \cdot \delta r})^2}, \text{ for } r \geq \frac{h_a}{\delta r} & C_k = \sqrt{A_k^2 + B_k^2} \\ z = \xi(\beta(m - m_1) + (n - n_1)) & \varphi_k = \arg(A_k, B_k) \\ \psi_k = k \begin{cases} 0 \text{ if } z \geq 0 \\ \pi \text{ if } z < 0 \end{cases} & \Omega_C = 2.\omega_c v_a . \cos(\theta)\sin(\phi)/c \end{array}$$

$A(\phi)$ is the backscattered wave by a clutter patch at azimuth ϕ and unit of time. This term aggregates several physical parameter, such as the pulse power, the backscattered Radar Cross Section of the patch, the EM losses for propagating in the air and obviously the antenna pattern. Second order (i.e Bragg) backscattering depends mainly on the elevation angle θ. For this reason, an identical electromagnetic backscattering over the isorange/isoelevation circle, i.e. $A(\phi)$ constant, that is an isotropic clutter model, is realistic. A clutter model, for $A(\phi)$ varying with ϕ, such as the sea clutter for instance, is out of the scope of this work, but would not drastically change our results. For the chirp waveform with range compression, the weighting $\mathrm{sinc}^2(\Omega_C.\tau/2)$ is exact for monochromatic wave but it is an approximation for chirps, valid with the radar parameter exposed in section 4.4. Moreover, we also assume that the backscattered energy remains constant during the pulse illumination, i.e. the backscattering is not frequency dependent, in order to achieve an effective bandwidth equal to B and thus an effective resolution equals to $\delta_r = c/2B$ (see [99] for range defocusing problems).

Radio Frequency Interferences (RFI)

Unlike the jammer disturbance, the RFI are generally unwilled emissions within the radar band by another electromagnetic source (e.g. for radio-communication purpose).

This source is spatially located in an azimuth and elevation direction denoted ϕ_{RFI} and θ_{RFI} respectively, and although fixed it exhibits a Doppler f_{Dopp} component to fit with the aircraft context. The RFI are characterized by quiet periods of random duration T_q and strong emission (bursts) of random duration denoted T_b (see [100]). During an emission, the received voltage from a RFI (i.e. the RFI power), although random, remains constant (and then it is considered as equal for all the pulses for which the RFI occurs unlike the Jammers as seen below). Thus, the RFI covariance matrix entries can be derived from Eq. (4.14), where $J(t-t_1, m, m_1)$ depends on the power of the RFI P_{RFI}, on the random quiet and burst periods as well as on voltage RFI, brief details are given in List of Tables. For a chirp waveform the covariance entry are given by:

$$\begin{aligned} r_{(n,m),(n_1,m_1)} = & P_{RFI} e^{-j(n-n_1)\pi \cos(\theta_{RFI}) \sin(\phi_{RFI})} \\ & e^{2j\pi(m-m_1)\frac{f_{Dopp}}{f_r}} \\ & \int_{-\tau/2}^{\tau/2} \int_{-\tau/2}^{\tau/2} e^{j(\omega_c + K_\omega t)t} e^{-j(\omega_c + K_\omega t_1)t_1} J(t-t_1, m, m_1) dt dt1 \end{aligned} \tag{4.14}$$

Where:

$$f_{Dopp} = v_a \frac{2}{\lambda} \cos(\theta_{RFI}) \sin(\phi_{RFI})$$

Jammer

Finally, for the jammer model, we use a simple barrage jammer model. In other words, it is a strong emission over the whole radar frequency band and for all the backscattered pulses. For the sake of simplicity the received voltage is considered independent from one pulse to another, but the signal properties as constant over the observation time. Moreover this emission is spatially located in azimuth and elevation direction denoted ϕ_J and θ_J, respectively. Thus, the covariance matrix is computed from Eq. (4.15), where P_J is the power of the signal received from the jammer, which follows a Rayleigh distribution.

$$r_{(n,m),(n_1,m_1)} = P_J \tau^2 \operatorname{sinc}^2(\omega_c.\tau/2) e^{-j(n-n_1)\pi \cos(\theta_J) \sin(\phi_J)} C(m-m_1) \tag{4.15}$$

Where $C(m-m_1)$ is a function which associates a random complex number for each value $m-m_1$, with $|C(m-m_1)| = 1$ and $\arg(C(m-m_1)) \sim U(0, 2\pi)$. As with the clutter model, the weighting $\operatorname{sinc}^2(\omega_c.\tau/2)$ is exact for monochromatic wave but it is an approximation for chirps, valid with the radar parameters exposed in section 4.4 (see List of Tables).

Fig. 4.3 shows an example of a synthetic signal distribution in the space-Doppler

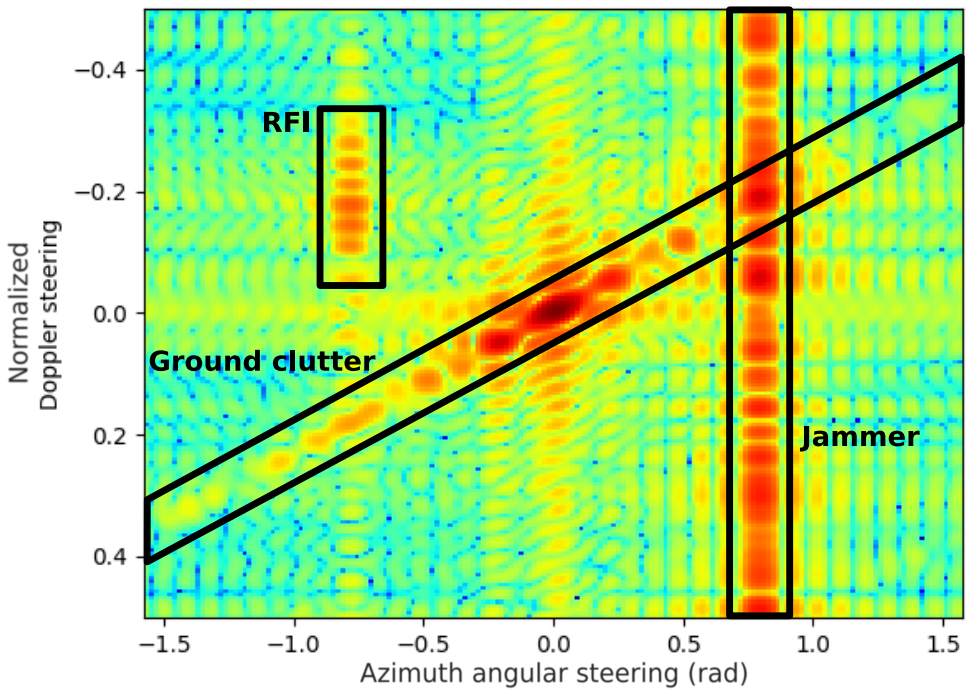


Figure 4.3 – Example of synthetic noises power generated from equations defined in Sec. 4.2.1.

domain where we observe Gaussian noise, jammer, clutter and RFI. On the right part of Fig. 4.3, we see the jammer signal which is highly correlated in the space domain and can be filtered using only the spatial dimension of the STAP filter. On the left part of Fig. 4.3, the RFI is also highly correlated in the spatial domain but can be filtered more efficiently if the STAP filter also uses the Doppler domain, in particular when the target comes from the same direction. On the diagonal of the image, the clutter exhibits correlations along a ridge whose slope can differ depending on the aircraft speed, so the optimal filtering dimensions will be different for two distinct clutters. The Gaussian noise creates a background power which is not clearly visible in Fig. 4.3 because its power is several orders of magnitudes under the interference power.

We can see on simple noise and interference models that the sample matrix observes different patterns related to the properties of each disturbance. Fig. 4.2 shows a simplified example of the patterns that can be recognized and linked to a type of noise (details are given in the next section).

Optimizations

Due to the large amount of data to be generated in the learning step, as shown in Section 4.2.3, the generation of the disturbance matrices has to be optimized. The noise and clutter generation can be time consuming, but the models described in this section can be efficiently implemented. Three optimizations are performed:

1. The matrix is Hermitian. With this hypothesis, it is possible to compute $\mathbf{R}_r$ only for $(n - n_1) \geq 0$ and $(m - m_1) \geq 0$.
2. Some computations which depend on $(n - n_1)$ and $(m - m1)$ can be pre-computed in lookup tables. Indeed, because $n, n1 \in \{0, ..., N-1\}$ and $m, m_1 \in \{0, ..., M-1\}$, there are only $2N - 1$ values for $(n - n_1)$ and $2M - 1$ values for $(m - m_1)$. This is related to the block Toeplitz property of the covariance matrix.
3. Parts of the equations are independent and can therefore be computed in parallel.

4.2.2 Dimension Reduction Rules

Each covariance matrix must be associated to a couple (N, M) for the learning phase. First, the following rules are used to select the N/M ratio:

R1 For clutter, the N/M ratio is deduced from the clutter ridge angle in the space-Doppler plan.

R2 For jammer, $N = N_f$ and $M = 1$.

R3 For RFI, we fix an empiric ratio where $N/M \approx 9/4$ since this type of noise is mainly spatially correlated but can also be distinguished from its Doppler correlation.

R4 If several interferences are present, the ratio N/M is fixed as the mean of the interferences ratio.

R5 If only Gaussian noise is present, the simple space-time matched filter is used instead of STAP (i.e. $w = s$, $IF = N_f M_f$ with an operational complexity of $\mathcal{O}(N_f M_f)$).

Once the ratio has been fixed, the NM dimensions are selected by taking the closest ratio available in the solution space (except for **R5**). These rules allow the validation of our concept but can be refined again. The best method for real-case applications would be to perform an exhaustive test of all (N, M) couples for every input noises and to select the best one regarding a loss function depending on the improvement factor and the computing complexity.

Another constraint, not included in the learning rules is that the predicted dimension values have to verify $NM < Q$ as stated in section 4.1. However, since the database does not contain matrix sample with $NM > Q$ (by the learning database construction as seen in the next section), it is very infrequent that this condition is violated for prediction step (less than 1% of our results). When this situation occurs, the larger value between N and M is truncated so that $NM \leq Q$.

4.2.3 Covariance matrix dataset generation

As usual when using machine learning, the learning dataset has to be carefully designed, even when the data is synthetic.

Input variables selection

A crucial point of the design of a good dataset is the selection of the input variables. In our case, this corresponds to the covariance matrix at the network input. A straightforward approach would be to use the dimensions $N_f M_f$ for prediction and to perform the dimension reduction from the full dimension matrix with the predicted dimensions NM. However, this approach has two drawbacks. The first is the great number of inputs which requires a large Artificial Neural Network (ANN) to fit the data during the offline process and leads to a higher computing complexity and a more difficult training[101]. The second drawback is the huge amount of ancillary data required to estimate $\tilde{\mathbf{R}}_r$ (as stated in section 4.1) during the online process. A most refined approach is to use already reduced matrices of dimension N_I and M_I for the ANN learning and for the online optimal matrix selection. These reduced matrices do not necessarily contain all the interference statistical properties as seen in section 4.2.1 but are large enough to select the optimal dimensions (i.e. N and M), which contain all this information.

Completeness

We also generate the most complete dataset possible to anticipate all situations that may occur. The signal can contain target returns and several interferences. The number of interferences $nb \leq 3$ (i.e. noise/clutter/jammer/RFI) is picked up with the probabilities $P(nb{=}0) = 0.1$, $P(nb{=}1) = P(nb{=}2) = P(nb{=}3) = 0.3$. These interferences are randomly picked up with the probabilities $P_C = 0.5$, $P_J = 0.25$ and $P_{RFI} = 0.25$ for clutter, jammer and RFI, respectively. This choice is meant to increase the weight of the clutter model

that comes with more diverse cases. If the ground clutter is selected several times, the duplicates are removed. The selected interferences are applied to several range gates and Gaussian noise is always added to the signal. Algorithm 1 describes the process that select noises.

Algorithm 1 Choice of interferences

```
inter_list ← {}
(P_C, P_J, P_RFI) ← (0.5, 0.25, 0.25)
Pick a random integer nb ∈ {0, 1, 2, 3} with probabilities (0.1, 0.3, 0.3, 0.3)
while nb > 0 do
   Pick a random integer inter ∈ {0, 1, 2} with probabilities (P_C, P_J, P_RFI)
   // 0: Clutter, 1: Jammer, 2: RFI.
   Push inter in inter_list
   nb ← nb − 1
end while
// Only one ground clutter can be present.
Remove duplicates 0 in inter_list
return inter_list
```

Once the interferences parameters are picked up, the corresponding matrix $\mathbf{R}_r$ of size $N_f M_f$ is created using the models described in Section 4.2.1. Synthetic measurements $\mathbf{y}_r$ are created for several range gates from $\mathbf{R}_r$ and the random coefficient array $\mathbf{u}$ (see Eq. (4.3)). A covariance matrix $\tilde{\mathbf{R}}_{r,p}$ of size $N_I M_I$ is estimated from these synthetic measurements with the sample matrix method described in Section 4.1 (we are close to the operational conditions). Then, each matrix $\tilde{\mathbf{R}}_{r,p}$ is associated to optimal NM dimensions according to the rules described in Section 4.2.2. Then, each dataset sample is composed of an input: a matrix $\tilde{\mathbf{R}}_{r,p}$, and two outputs: the optimal N and M dimensions, used both in the learning step and the prediction step as detailed in the next section.

4.3 Optimal dimension learning and prediction

In this section, we first detail the models used to predict the optimal matrix dimension for STAP processing. Then, we discuss our choice of the size of the reduced matrix dimension serving as input of the prediction models.

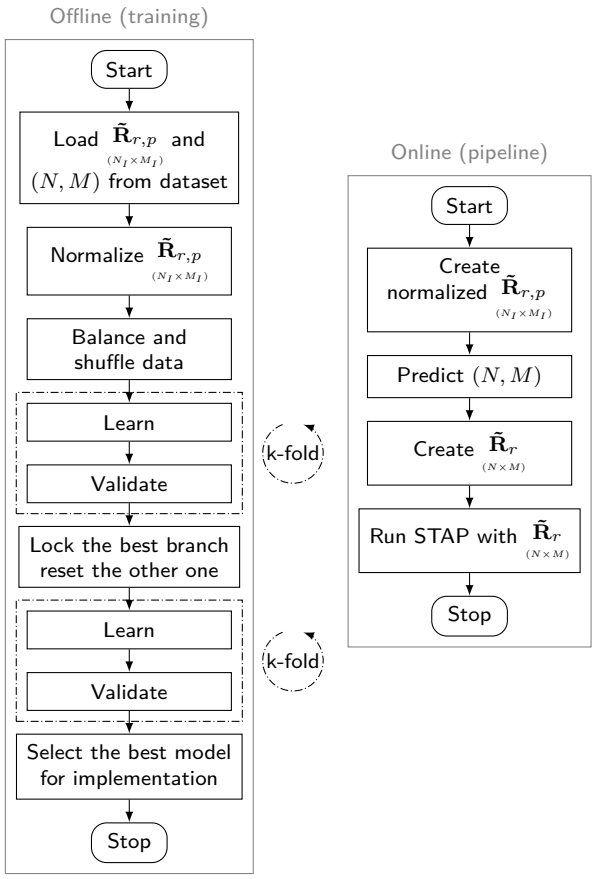


Figure 4.4 – Flowchart of the STAPLE algorithm

4.3.1 Models definition

To solve this prediction problem, we train a machine learning process to take full a advantage of the available knowledge held by $\tilde{\mathbf{R}}_{r,p}$. Moreover, the possible predicted couples (N, M) are constrained such that $NM \leq Q$ as stated in section 4.1. We choose to use only odd values for the dimensions.

We propose two models: the former is based on a CNN, and the latter combines a CNN and a RF. These two models have two different branches to separately predict N and M as outputs. Figure 4.4 left flowchart presents the training phase. The dataset generated as described in Section 4.2.3 is loaded and the inputs are normalized (i.e. each input is scaled over the interval $[-1, 1]$). The dataset is balanced by replication of the under-represented samples and is shuffled. K-fold cross-validation is used to check the models ability to generalize on different datasets; that is, the dataset is divided into k equal sized partitions and k training are performed sequentially on k-1 partitions and is

Table 4.2 – Learning setup

Neural network		Random forest	
Framework	Keras	Framework	Scikit-learn
Loss function	Huber	Loss function	MSE
Datatype (training)	float32	Datatype (training)	float32
Datatype (test)	float16	Datatype (test)	float16
Cross-validation	6-fold	Cross-validation	6-fold
Optimizer	Adam	Number of tree	50
Epochs	200	Max depth	100
Batch size	32		
Learning rate	0.001		

tested on the remaining partition. The trained algorithm which obtains the best results is kept. A second training starts after locking weights on the shared part of the model and on the branch which performs the best (branch N or M). This second training also runs with k-fold cross-validation. After this second pass, we keep the model that performs the best. The macro-parameters used for learning are summarized in Tab. 4.2. We present the parameters that worked best for our application with a reasonable computing complexity.

Model 1 (CNN)

A fully connected neural network holds a huge complexity, which would limit the optimisation offered by our solution. Therefore, we choose a 2D convolutional first hidden layer to reduce the computational load in the online processing. This layer performs a 2D filtering of the input data by learned kernels. It is composed of 8 triangle-shaped kernels of size 6×2 (2 for real and imaginary parts). For the sake of clarity, Fig. 4.5 shows how the convolution layer extracts features for the rest of the network.

The network then splits into two independent branches for the estimation of N and M. The prediction being performed independently for dimensions N and M, an error in the prediction of N does not imply an error in the prediction of M. Therefore, this seems a better option than the prediction of a couple (N, M) by a single classifier. The two first

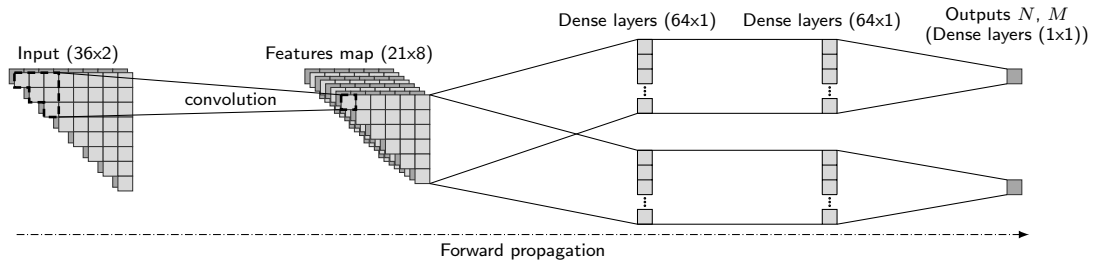


Figure 4.5 – CNN-based approach (Model 1)

layers of each branches are fully connected. The third layer of each branch is composed of a single neuron fully connected to the previous layer. This last layers has a single outputs which is trained with a regression method (with a Huber loss function) to give the required dimension for N and M. For both offline and online processes, the output of the network is a floating point value which is rounded to integer to dimension the STAP filter.

Fig. 4.5 presents the resulting neural network which holds the gain on complexity described by Eq. (4.16).

$$G_{\mathcal{O}} = \frac{N^3M^3 + kc + 2\sum_{i=1}^{L} n_i n_{i-1}}{N_f^3 M_f^3} \tag{4.16}$$

Where:

L: the number of layers per branch

k: the number of connections of one convolution kernel

c: the number of convolution outputs

n_i: the number of neurons on layer i

Model 2 (CNN + RF)

A common practice to reduce prediction variance in machine learning is to implement redundancy. Instead of having a unique prediction system trained on a dataset, it is possible to implement more neurons or branches and train them on different subsets of data. In neural network, it is possible by using pruning (with more neurons) or model averaging (multiplication of branches/models). However, redundancy in neural networks has a high computational cost. Hence, we propose to use random forests, which exploit model redundancy with a low computational effort. The proposed model is as follows: The

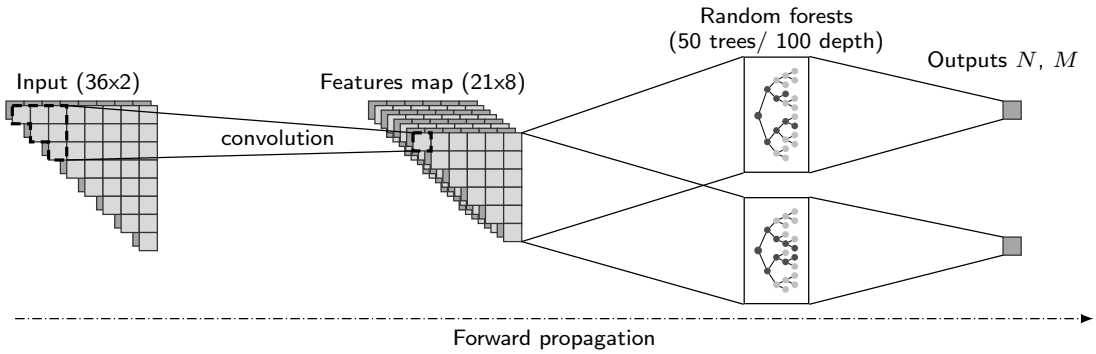


Figure 4.6 – CNN + random forest approach (Model 2)

model described in Section 4.3.1 is trained to extract features after the convolutional layer. If the neural network can extract features with the convolutional layer, then a random forest should be able to extract information as well. We then cut the neural network after the convolutional layer, and we connect a random forest that learns with the same dataset. Fig. 4.6 shows the model from Fig. 4.5 modified to replace the last layers by random forest predictors. Results from the neural network and this hybrid prediction model are compared in Section 4.4. Equation (4.17) shows the complexity gain of this second model.

$$G_{\mathcal{O}} = \frac{N^3M^3 + kc + 2td}{N_f^3 M_f^3} \tag{4.17}$$

Where:

tn: the number of trees

d: the trees maximum depth

4.3.2 M_I and N_I values

Initially, our idea was to empirically determine the optimal dimension, by testing several values under the assumption $M_I = N_I$. It was obvious that these dimensions have to be strictly higher than 2 in order to identify patterns within the estimated covariance matrix. We find that the values $N_I = M_I = 3$ lead to optimal values to perform this selection as detailed in section 4.4.1. For this reason, all the results presented in section 4.4 are based on these values. Thus, for $M_I = N_I = 3$ the top triangle, excluding the diagonal, is composed of complex numbers, for a total of $2(9 \cdot \frac{8}{2}) = 72$ values. The input

size of our CNN is therefore 72. Testing highest values would not improve our prediction results, but would strongly increase the computational load.

4.3.3 Prediction and filtering

Once the model is trained, we can use it to enhance the STAP. Figure 4.4 right flowchart shows the processing that must be performed online. The matrix $\tilde{\mathbf{R}}_r$ is created and the model predicts dimensions (N, M). We create the covariance matrix $\tilde{\mathbf{R}}_r$ with the predicted dimensions and run the STAP with $\tilde{\mathbf{R}}_r$. This algorithm is direct and thus can be pipelined to cover the computing cost of our prediction model.

4.4 Results

In order to simulate the covariance matrix estimate, we use the radar parameters $\tau = 20\,\mu s$, $T_r = 1000\,\mu s$, $\frac{\omega_c}{2\pi} = 5\,\text{MHz}$, $\frac{K_\omega}{2\pi} = 0.05\,\mu s^{-2}$. The plane parameters are picked from uniform distributions so that $100\,\text{m}\,\text{s}^{-1} < v_a < 1000\,\text{m}\,\text{s}^{-1}$ and $500\,\text{m} < h_a < 1500\,\text{m}$. In fact, while the radar parameters are generally fixed the operational conditions change and for the sake of reality, we randomly chose them. The interferences power, P_J and P_{RFI} are picked following Rayleigh distributions. The numerical values used to generate the interferences are reported in Table 4.3. In our case study, we use the arbitrary value of $Q_1 = 170$. The limit Q_3 is fixed by the available number of antennas and slow-time cells: $N_f = M_f = 35$. Thus in our case, the predicted values have to verify $MN < Q = 170$ (the limiting criterion being the complexity). Figure 4.7 shows the possible (N,M) solutions which are such as $N \in [1, 35]$, $M \in [1, 35]$ and $NM < 170$. In the next two sections, we detail first the results of the optimal dimension prediction for the two models (CNN and CNN+RF) while the second section is focused on the STAP performance metrics previously defined. In particular, the comparison is made with respect to the full STAP performance but also to stand reduction dimension approach involving choosing a value $N = M$ such as the values verify $MN < Q$.

4.4.1 Dimension prediction performance

We further reduce the number of solutions by taking the envelope of the solution space. To show that the designer can choose different compromises (i.e. the upper bound of Q) between quality of results and computing complexity, we use two solution sets for the case

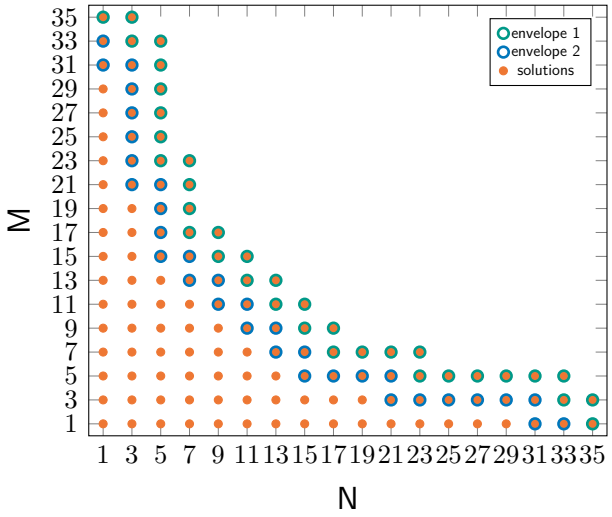


Figure 4.7 – Solutions space of NM dimensions.

Table 4.3 – Test values

τ / T_r	20 µs / 100 µs
$\frac{\omega_c}{2\pi}$ / $\frac{K_\omega}{2\pi}$	1 GHz / 0.05 μs^{-2}
$\mathbb{E}[P_J]$ / $\mathbb{E}[P_{RFI}]$	1 GW/15 kW
N_f / M_f	35 / 35

study. The first, referred to as 'envelope 1' in Fig. 4.7, is constituted of the frontier of the solution space. The second, referred to as 'envelope 2' in Fig. 4.7, is composed of the frontier of the solution space after exclusion of the solutions in envelope 1.

The training of the models is performed on a dataset of 10000 samples created as described in Section 4.2. We balance the dataset by replicating data for the underrepresented N and M values and shuffle the resulting dataset before learning. During the two k-fold cross-validation, we observe correct results on each subsets, which means that the models can be generalized on different datasets. Equations (4.18) and (4.19), derived from Eq. (4.16) and (4.17), show the complexity gain and bounds (for envelope 1) for the CNN model and the CNN + RF model, respectively.

$$G_{\mathcal{O}} = \frac{N^3M^3 + 3.2\text{e}4}{1.8\text{e}9}; \quad -47.5\text{dB} \leq G_{\mathcal{O}} \leq -25.7\text{dB} \tag{4.18}$$

$$G_{\mathcal{O}} = \frac{N^3M^3 + 1.2\text{e}4}{1.8\text{e}9}; \quad -51.7\text{dB} \leq G_{\mathcal{O}} \leq -25.7\text{dB} \tag{4.19}$$

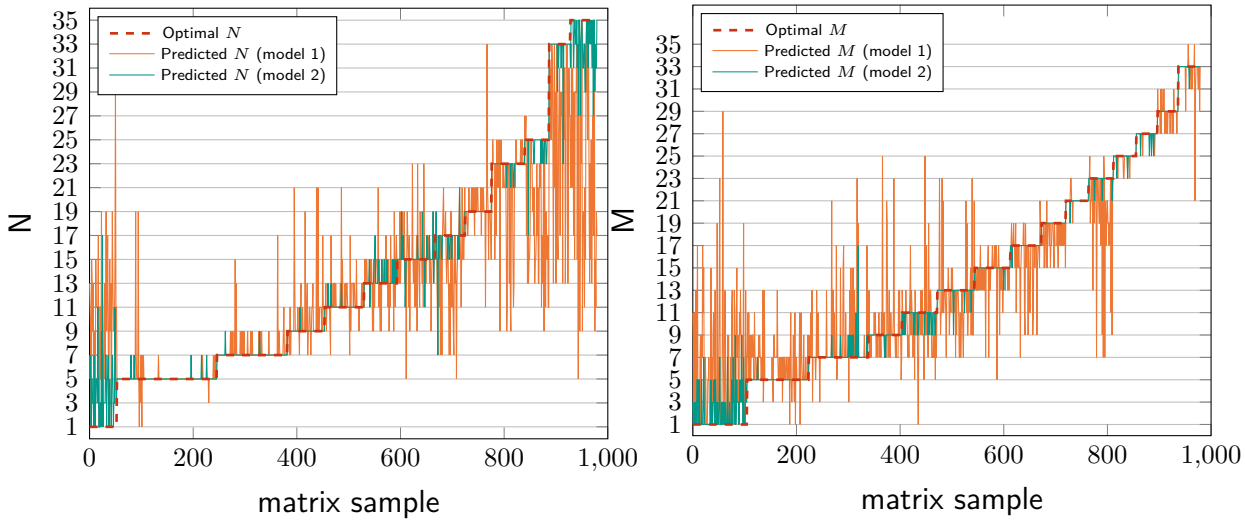


Figure 4.8 – Prediction of N and M on a random test set (ordered).

Figure 4.8 shows the result of predictions on a random subset of 1000 samples (reduced subset for the sake of a correct visualisation) for model 1 and model 2. We observe that the model 1 is able to predict the correct value in most case but the prediction accuracy is not good. However, model 2 prediction performs generally very well and most errors are small. For this reason, we only detail the second model results in the following. Figure 4.9 shows the complexity of the different versions of the STAP filtering and of the prediction algorithm. We see that a lot of complexity is saved by our method over the $(N = M)$ approach. Moreover, the complexity of our algorithm is negligible compared to the full STAP and to the $STAP_{13\times13}$. However, this complexity is high compared to the predicted complexity in the situation where STAP is not required $(N = M = 1)$.

Fig. 4.10 shows the confusion matrix for the branch N and M for the CNN + RF model. The true values to predict are on the left column. The cells show the ratio of predicted values for the true value of the corresponding row. The last column of the matrices gives the number of sample for the corresponding value of N or M. These results show that the prediction algorithm performs well on most data. However, the algorithm is not efficient when N is close to 1. We can also see that some (N, M) couples are absent from the dataset. This probably comes from the realistic aspect of our dataset and the rules defined in Section 4.2.3 which limit the possible configurations. We observe that even when the predicted value is not exact, it is nevertheless generally close to the expected value.

We noticed that even though an increase of the model complexity slightly improves

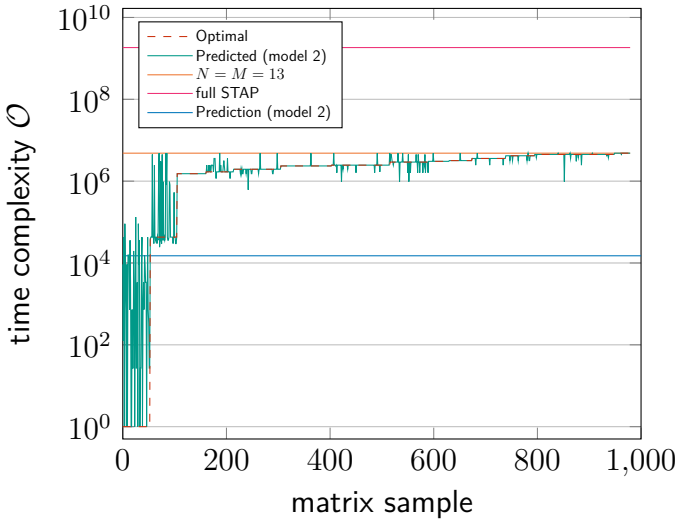
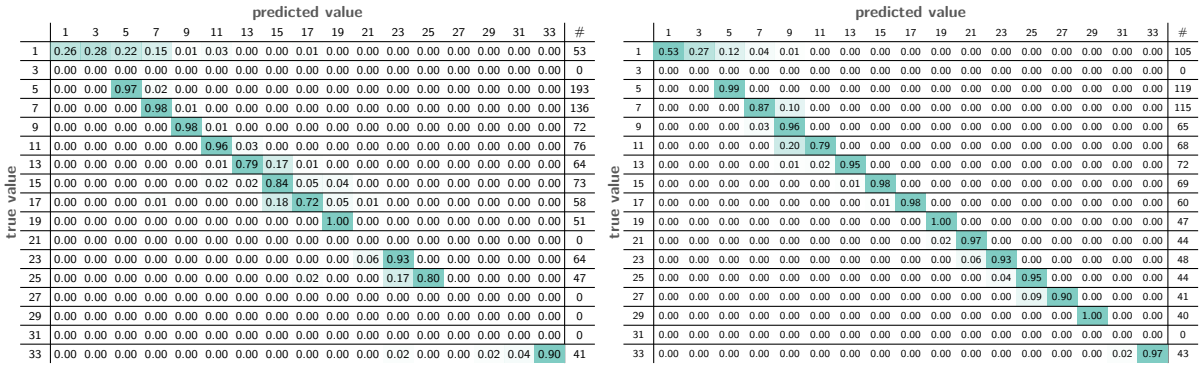


Figure 4.9 – Computing complexity on a random test set (ordered).



predicted value

true value	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	#
1	0.26	0.28	0.22	0.15	0.01	0.03	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	53
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
5	0.00	0.00	0.97	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	193
7	0.00	0.00	0.00	0.98	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	136
9	0.00	0.00	0.00	0.00	0.98	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	72
11	0.00	0.00	0.00	0.00	0.00	0.96	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	76
13	0.00	0.00	0.00	0.00	0.00	0.01	0.79	0.17	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	64
15	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.84	0.05	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	73
17	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.18	0.72	0.05	0.01	0.00	0.00	0.00	0.00	0.00	0.00	58
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	51
21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.93	0.00	0.00	0.00	0.00	0.00	64
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.17	0.80	0.00	0.00	0.00	0.00	47
27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02	0.04	0.90	41

(a) N confusion matrix

predicted value

true value	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	#
1	0.53	0.27	0.12	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	105
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
5	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	119
7	0.00	0.00	0.00	0.87	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	115
9	0.00	0.00	0.00	0.03	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	65
11	0.00	0.00	0.00	0.00	0.20	0.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	68
13	0.00	0.00	0.00	0.00	0.01	0.02	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	72
15	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	69
17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	60
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	47
21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.97	0.00	0.00	0.00	0.00	0.00	44
23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.93	0.00	0.00	0.00	0.00	0.00	48
25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.95	0.00	0.00	0.00	0.00	44
27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.90	0.00	0.00	0.00	41
29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	40
31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.97	43

(b) M confusion matrix

Figure 4.10 – Confusion matrices of the CNN + RF model for 1000 samples (%)

the prediction, it does not bring a significant accuracy gain. However, the key for a better prediction accuracy may be a better estimation of the optimal dimensions. Indeed, the rules exposed in Section 4.2.2 take into account the interference angle but not other aspects like the relative power of each interference present in the signal. This may make regression more difficult for the neural network. This was confirmed in tests on a smaller dataset where each interference had the same power, on which we obtained a prediction accuracy of over 97%.

The results of the prediction are used to create the appropriated covariance matrices, and the performance are compared with the $N = M$ STAP filter, with the same constraint on Q_1 as detailed in the next section.

Table 4.4 – Performances of the static and the adaptive method (dB ratio w.r.t. full STAP).

	$G_{\mathcal{IF}}$	$G_{\mathcal{CN}}$	$G_{\mathcal{O}}$
$STAP_{13\times13}$(dB)	-9.04	-8.61	-25.8
$STAPLE_1$(dB)	-7.09 ▲	-9.35 ▼	-28.8 ▼
$STAPLE_2$(dB)	-7.69 ▲	-10.7 ▼	-35.1 ▼

4.4.2 STAP performance Metrics

Table 4.4 shows the ratio of the IF, the condition number of the matrix and the computational load of the standard $N = M$ approach ($STAP_{13\times13}$, so that $NM < Q$) and our adaptive system ($STAPLE_1$ and $STAPLE_2$ for envelope 1 and 2 from Fig. 4.7, respectively) compared to the full STAP. The IF and the computing complexity ratio are computed from the mean dB ratio and the condition number ratio is computed from the median dB ratio to cope with extreme values of κ which might appear when $\lambda_{min} \to 0$.

As expected, we observe a lower improvement factor for all the reduced dimension algorithms compared to full STAP ($G_{\mathcal{IF}} < 0$), which is balanced by gains on other metrics ($G_{\mathcal{CN}}$ and $G_{\mathcal{O}}$). Comparing now, our approach with the usual/rough dimension reduction ($N = M = 13$), $STAPLE_1$ and $STAPLE_2$ offer a mean improvement of 1.95dB and 1.35dB (difference between the second and first line in table 4.4 for $G_{\mathcal{IF}}$, and difference between the third line and the first line) respectively. Thus our approach outperform the disturbance rejection of the usual dimension reduction approach.

For both standard and adaptive reduction dimension approaches, the measured matrix condition number is lower than that of the full STAP, thus matrix inversion is strongly more precise in operational conditions for these two approaches with regards to the full STAP. Comparing again the standard reduction with our approach, the improvement of the condition number is much higher for STAPLE than for the standard approach (-0.74dB and -2.09dB) (difference between the second and first line in table 4.4 for $G_{\mathcal{IF}}$, and difference between the third line and the first line) showing that our adaptive dimension reduction provides more stable matrix inversion and thus more stable effective filtering in operational conditions.

However the main improvement of our approach is for the computational load. computational load of the reduced solutions is also way lower than with the full STAP. This allows realistic implementations of the algorithm. $STAPLE_1$ and $STAPLE_2$ allows the reduction of -3.0dB and -9.3dB, respectively, on the computational load compared to

the standard approach. This means that two to more than eight times less computations are required to obtain a filtering quality better than the $N = M$ approach. Consequently, the radar system can perform up to eight times as many scans than the static approach in the same time frame. This gain is observed for our specific application and is expected to be even more significant if used on larger systems. The selection of the solution set (e.g. envelope 1 or envelope 2) allows to choose a compromise between the quality and complexity of the calculation but must be set manually.

The correctness of N and M prediction limits the loss of filtering quality of our reduced STAP algorithm while reducing the processing complexity. We observe that the computational cost of the prediction algorithm is negligible compared to the STAP itself and thus saves a lot of processing time. Moreover since our approach is based on physical models allowing us to derive interference statistical properties, i.e. covariance matrices, we can expect that the result improvements, in particular the computational load reduction, are not affected by the radar/plane/RFI/Jammer parameters. Possibly, others empirical values, such as that of rules three in section 4.2.2, could be refined in order to improve the results, but this optimization is out of the scope of our work.

4.5 Conclusion

This work presents STAPLE, a new method which reduces the dimension of STAP by inferring the optimal filtering dimensions at runtime from a reduced interferences covariance matrix. To the best of our knowledge, this is the first method which successfully uses machine learning to select STAP dimensions at runtime. We have defined several models of clutter interference to test this concept. The results show that STAPLE is a promising solution for the implementation of efficient reduced STAP, for both computational complexity reduction and quality the filtering and then target detection improvement. Through the use of learning algorithms, STAPLE can be adapted to a wide range of radar applications and STAP techniques with little effort. Hence, this method of STAP adaptation to interference correlations in the space-Doppler plan is particularly relevant in aircraft systems where the computing system has strong limitations and evolves in an uncertain environment. Indeed, this method does not rely on assumptions on the aircraft environment to be efficient (e.g. aircraft speed to locate the clutter ridge). This method may require a lot of data to be applied to a specific application. If measured data is not available in large quantities, the available data may be enriched with synthetic signal.

Like most learning techniques, the quality of results is highly dependant on the quality of training data. Experts must give special attention to the creation of their dataset to benefit from STAPLE. More specifically, the selection of the NM dimensions for a given covariance matrix depends on a tradeoff between quality of results and computing complexity which is application specific. This work allows us to consider dedicated hardware reconfigurable architectures, in order to fully benefit from the acceleration potential of STAPLE.

In Chapters 3 and 4 we presented adaptive systems which reconfigure themselves according to the current context to improve performance and quality of results. These optimizations are only possible because they embed application knowledge. This knowledge belongs to application experts, who do not have the skills to implement the solutions in hardware and/or software. Hence, to design adaptive systems, we need a methodology that ease the communication between hardware, software and application experts. The goal is to have these experts to efficiently work together to get the best of all worlds. In the following chapter, we propose such a methodology to enable a heterogeneous team of experts to design an embedded adaptive system in a SoC FPGA.

Chapter 5

Methodology for reconfigurable systems design

CANDID: Collaborative Agile desigN framework for aDaptIve Dataflow architectures

Contents

As the implementation of dynamic partial reconfiguration (DPR) of FPGA is getting easier and the use of SoC architectures increase, we observe that DPR is not yet widely used in industrial projects. Although some researchers blame the long reconfiguration time, this time is getting lower and can be seamless if some strategies are well employed. The reluctance of DPR use in Industry may come from the difficulty to design efficient and safe adaptive systems. The application knowledge is mandatory to create efficient reconfigurable systems, and this knowledge often belongs to experts who are unaware of underlying architectures. Hence, they require high level definitions of the reconfigurable application and decisions. However, while many academic and industrial tools support the implementation of reconfiguration at low level (Vivado, GoAhead, IMPRESS, etc.), there is yet no tool to support a high level definition of these adaptive systems. Hence, this chapter presents CANDID, a framework for the design and development of application driven reconfigurable systems. This methodology can be viewed as a multi-agent process, whose agents can be various specialists and/or AI. This methodology is designed to be compatible with the Agile development paradigm. This chapter also describes a set of tools to support this methodology. Eventually, we present a case study which uses this methodology to create an embedded reconfigurable radar tracking system.

5.1 Introduction

Reconfigurable computing has been a research subject for several decades. Many models, methods and tools have been proposed for the design of architectures that can efficiently be reconfigured to implement multiple different specialized accelerators with the same limited hardware resources. We observe the emergence of many reconfigurable architectures, that rely on two main categories: coarse-grained reconfigurable array (CGRA) and reconfigurable FPGA. The CGRA are easier to program and faster to reconfigure, but they are less flexible than FPGA which are fine-grain reconfigurable architectures. Despite first attempts 20 years ago, few commercial products and tools are available for CGRA compared to FPGA which benefit from decades of experience. However, FPGA are costly to develop, and the reconfiguration time used to be prohibitive for runtime adaptation. The support of reconfiguration was also poor and developing these architectures was error-prone and time consuming. Nowadays, the reconfiguration time has been lowered and the support for DPR improved. Recent tools benefit from the introduction of reconfigurable design flows in vendors tools, and the creation of software driven reconfiguration

(e.g. through the processor configuration access port (PCAP)) with its associated application programming interface (API). Furthermore, several academic tools were proposed to improve the support of DPR by supporting hierarchical DPR and offering multi-grain reconfigurations to improve flexibility and to offer faster reconfiguration [71], [72].

Yet the use of DPR does not seem to spread in industrial projects. This reluctance may come from the difficulty to design efficient and reliable reconfigurable architectures at a reasonable cost with current Industry development processes. Indeed, considering execution time, power consumption as well as design and product costs is not sufficient. The quality of service (QoS) is of course of prime importance. Nevertheless, the application knowledge is mandatory in QoS-based reconfiguration for two reasons. The first one is to know which parts of the processing systems can be reconfigured, and the second is to find relevant QoS criteria. The application knowledge belongs to experts which are not necessarily aware of the underlying implementation of the system and by extension of the reconfiguration process. As a matter of fact, the development of efficient adaptive systems requires a variety of experts with different skills in application, hardware (HW) and software (SW) who must cooperate. The analysis of current practices and available tools lead to the conclusion that a new global methodology is actually required to guarantee the separation of concerns between multiple agents who must collaborate according to their respective expertise and skills.

In addition to these considerations, we believe that this methodology must be compliant with the Agile development approach which is increasingly used in industrial projects [102] and well adapted to the nature of such complex adaptive systems that can benefit from an iterative design process. Agile concept was first used for SW development, but as the abstraction level of HW development rises (e.g. with the HLS tools) the same principles can be applied to HW[103]. To be compatible with Agile approaches, our methodology must allow to modify the system design after the first development. Therefore, the framework must allow incremental design with minimal overhead. Furthermore, there is no hierarchy in an Agile development team, so we must consider that the difference between agents is only based on their respective skills.

We detail our methodology in this chapter which is organized as follows. Section 5.2 first presents a state of the art on QoS-driven adaptive architectures to expose the need for such architectures. It then provides a state of the art on CAD tools which support DPR. Section 5.3 first defines what is considered as an agent in our framework, and how we categorize agents to guarantee separation of concerns in the collaborative environment.

Then, we present the methodology to define the adaptive system. Section 5.4 presents the CAD implementations of our methodology. Eventually, Section 5.5 details a case study to show the process of developing a realistic radar tracking processing system with our framework.

5.2 State of the art

Architecture optimisation based on HW reconfiguration is mainly considered to enhance performances and reduce power consumption [32], [41], [43]. However the best HW/SW configuration may first meet the functional requirements. In [2], [3], [44] radar applications are reconfigured regarding various QoS criteria. In these studies, the optimization is not about switching between optimal or degraded algorithm but instead choosing the most adapted algorithm depending on the current situation. The reconfiguration criteria are diversified and domain-dependent, and only application experts can choose and formalize them.

Several tools were proposed to ease the implementation of reconfigurable systems (see [31] for a survey on DPR tools). Some tools enable the description of DPR at a description level which requires good HW skills, far from the usual knowledge of application experts [75]. DPR adds a layer of complexity to the already difficult task of hardware architecture design, and we see that efforts to facilitate the design of DPR architectures do not allow an application expert to effectively benefit from reconfigurable FPGA. In this work, we target the design of adaptive systems and therefore seek to rise the level of description of the reconfigurable system in order to make it accessible to application experts, while letting the implementation considerations to the HW experts. It means that our methodology must be compatible with most flows and tools for the final implementation.

Creating a design framework can be a difficult task with many considerations. In order to define the methodology and the associated tools, we need a way to model how the agents collaborate to create an efficient design. Because of the similarities between a cooperative design and the multi-agent systems, we use the definitions from the multi-agent domain to support our work.

5.3 Multi-agent methodology

5.3.1 Multi-agent design

5.3.1.1 Definition and specification of Agents

The design of many systems requires the cooperation of various specialists. This cooperation through a common methodology or CAD tools can be analysed as a multi-agent systems. The agents have different skills and can contribute at different points of the design process, with their respective skills and points of view. Some agents could even be AI, which learned to solve some design issues [104]. In Agile frameworks, the development team has no hierarchical organization but is multidisciplinary. The team must avoid dependence towards omniscient developer as much as possible. But of course, all the agents still have specialties. They can be application experts, HW experts or SW experts. Moreover, some experts may have hybrid specializations. The difference of specialization must be considered while developing a collaborative multi-agent system. The concerns of each agent is different, and this must influence the spectrum of actions and the interface of the different agents. Therefore, in Table 5.1a we propose to define the agents skills in the three main expertise domains and on two levels: developer for a level of knowledge good enough to develop a block in this domain, and architect for a level required to design the system at full-scale. Table 5.1b shows an example of specification of several agents skills.

5.3.1.2 Agents action space

Given the skills of each agent, they can contribute to specific parts of the design. We now define two types of contributions: observations and actions. Observations are issued queries about the modifications and decisions made in the domain they are observing. Actions are direct contributions/modifications related to their domains. Hence, agents can communicate directly through queries and answers and indirectly through their actions on environment. The following statements are true for each expertise domain:

1. The expert of domain A can issue observations for any domain B including A.
2. The expert of domain A, with developer skills can take actions on domain A development, and issue observations on domain A architecture.
3. The expert of domain A, with architect skills can take actions on domain A architecture, and issue observations on domain A development.

Figure 5.1 shows the actions spectrum of agents 2 and 3 of Table 5.1b.

Table 5.1 – Agents skills levels

(a) Association of skills and spectrum of actions

	Skills level	
Exp. domain	Developer	Architect
App.	Designs an app. block $^{\alpha}$	Designs the application ^{A}
HW	Designs a HW accelerator $^{\theta}$	Designs the HW architecture $^{\Theta}$
SW	Designs a piece of code $^{\sigma}$	Designs the SW architecture $^{\Sigma}$ (OS, scheduling)

(b) Example of agent specification (from Section 5.5)

	Agent 1		Agent 2		Agent 3	
Exp. domain	Dev.	Archi.	Dev.	Archi.	Dev.	Archi.
App.	✔		✔	✔		
HW	✔	✔			✔	
SW	✔	✔	✔		✔	

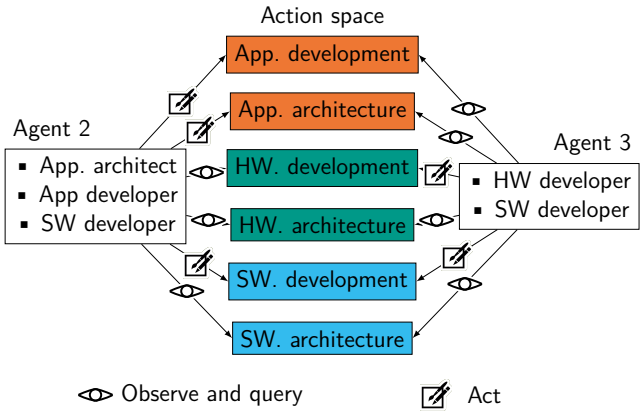


Figure 5.1 – Actions spectrum for two agents

5.3.1.3 Design philosophy

Our approach does not limit the solution space by anticipating possible further constraints. Thus it favors the generation of solutions that can be pruned later. A first example is the configuration controller that initially produces solutions that are correct but cannot happen. They will be pruned by an Application Architect who will introduce explicit banning rules. Another example is the possibility to propose a large number of configuration including impossible or costly mappings. These solutions will be easily eliminated by a HW Architect.

5.3.1.4 Working assumptions

To define a design framework we need to make assumptions to simplify a problem otherwise too large. We assume that:

- Targeted architecture is a SoC with a reconfigurable FPGA and a processor.
- Application can be modeled as a dataflow.
- The interconnect are of two types: wired dataflow and shared memory. Another option is a NoC which is not considered in this work.
- Self-organization capabilities of Agents mean they can synchronize their actions.
- Agents are aware of constraints specific to SoC design (e.g. limited memory bandwidth, conditional statements potential overhead).
- Agents are able to validate solutions they propose at every step of the design, through external tools/methodologies.

The industrial constraints impose strict reliability, so we consider the following limitations without loss of generalities:

- Scheduling is static and does not include DPR which can only occur between two configurations. Future work could include approach such as the real-time scheduling of DPR proposed in [105]. Such tool would be available for SW architects but would require the cooperation of HW architects since it requires HW/SW synchronization mechanisms.
- Partitioning of reconfigurable partitions (RP) is done manually. Solutions such as the one proposed in [106] could be used by HW architects when available at industrial grade.

Now that agents specialties and specters of actions are defined, we will categorize every action (with the tags from Table 5.1) to guarantee a good separation of concerns between experts.

5.3.2 Adaptive system definition

This section describes the required steps to define an adaptive architecture with our methodology. The different steps can overlap in time thanks to the communications between agents. As soon as enough information is available to start the next step, agents can begin to work on the next phase.

```
actor Name:
  inputs:
    type input_i[nbChannels_i]
  outputs:
    type output_j[nbChannels_j]
  sparameters:
    #Change at graph iterations
    #Parameter s_k belong to set E_k
    s_k ∈ E_k
  cparameters:
    #Change at actor iteration
    #Parameter c_l belong to set E_l
    c_l ∈ E_l
  tokens:
    input_i => nbTokens_i
    output_j => nbTokens_j
  description:
    Literal description of the actor
    computations.
end
```

Figure 5.2 – Description of a PiSDF actor.

Step 1 Static application definition

The first task is to define the application in a static form. Application experts are able to design a processing flow with a high level language such as Python or Matlab. They must describe the application as a block diagram, as it offers a natural representation for application experts [107, Chapter 2]. More specifically, we use the parameterized and interfaced dataflow model (PiSDF)[108] for the application definition. PiSDF is particularly adapted to model reconfigurable dataflow applications, such as radar, networking and imagery. To create the block diagram, the application expert must divide the processing into processing blocks (actors) and connections (arcs)[A]. The description granularity should be coarse at first place and can evolve during the project based on exchanges between experts (for example at Step 2 and Step 3). App. architects must define the important global constraints, such as required latency and throughput[A].

Table 5.2 – Specifications of a reconfigurable actor

Question	Answer type
Reconfigurable	Boolean
Always active	Boolean
MIB	Boolean
Configurations	List
QoS criteria	List
Actor description (Fig. 5.2)	Files list
Test vectors	Files list
HW implementable	Boolean list
SW implementable	Boolean list
Data parallelism	Boolean
Minimal version exists	Boolean
Minimal version	Config. name
Implem. strategy (Fig. 5.4)	Integer

Required at step: Step 3 | Step 4/Step 5 | Step 4

Step 2 Reconfigurability specification

Once the static system is well defined, we need to specify which actors can be reconfigured, and which blocks are not always active (as it is another kind of reconfiguration)^{A}. We can also tag some processes as 'multiple instances benefit' (MIB)$^{\alpha}$, which means that at least one instance must be launched, but if we have enough resources, we should run as many instances as possible. The team must then detail what are the different configurations$^{\alpha}$. This step is an application development action, because it takes place at actor scale and developers can suggest configurations because of performances concerns (algorithm complexity / quality of processing). The team must also specify which QoS criteria are available in the application, and in which actors we get them$^{\alpha}$. If the system is a sub-system of a larger one, external QoS criteria can be specified^{A}. Fig. 5.11 present a block diagram where the configurations, the MIB and QoS criterion are specified. Eventually, actors must be described (what computation is done, what are the input/output format and what are the valid set of PiSDF parameters)^{A}. Fig. 5.2 provides an example of an actor description. The QoS criteria computation description must be described in the actor description and an output port should be declared.

From these primary specifications, the developers must define the way that the system switches between configurations$^{\alpha}$. To simplify the specification of configuration changes, one controller per reconfigurable actor is defined. An additional controller can be defined to deal with the PiSDF configurations changes (to set the parameters defined in Fig. 5.2). Fig. 5.12 is a shared block diagram which represent how the application configuration changes regarding the QoS criteria. Once this part of the work is done, we can simulate the system to observe which states and combinations of configurations are reached. Some configurations combination can be eliminated if app. agents consider that it does not make any sense (an example is given in section 5.5)^{A}. The remaining combinations must be validated at application level (e.g. by Python simulations)^{A}. This validation must verify computation correctness at different checkpoints (between all actors), and provides the results as input test vectors for HW/SW experts. Table 5.2 formalizes all the specifications which must be made for every actors.

Step 3 Design space exploration

Once the system reconfiguration has been specified at application level, we need to define how the implementation is done. Fig. 5.3 presents several ways to implement the parametrization of the dataflow actors: with a configuration port (Strategy P1), or through the data channel (Strategy P2). This design choice is left to the discretion of the SW and HW architects and can evolve between development iterations. This specification must be made for the whole system (for HW and SW) to mitigate errors. We must now define what actors will run exclusively in HW or SW, and what will be able to run in both HW and SW. While HW/SW codesign is a well known problem, there is no perfect solution to explore the full design exploration space. But we have HW and SW experts, who can help to tackle this problem. In most application specific designs, the experts know that some actors can be implemented exclusively in HW, SW or both. A first step is to tag the actors that can run in HW or in SW from experts knowledge$^{\sigma\,\theta}$. This action must be done per application static actor, and per configuration for the reconfigurable actors. Now that the first exploration space reduction is done, we must do exploration for the actor we are not sure where they should run. The HW exploration can be done with the help of HLS tools (such as Vivado HLS)$^{\theta}$, while the SW exploration can be performed directly by running cross-compiled code on target processor using Xilinx SDK or Intel SoC FPGA EDS$^{\sigma}$. We can estimate the execution time and resources consumption of every actor which is not already fixed in HW or SW. These estimation data as well as

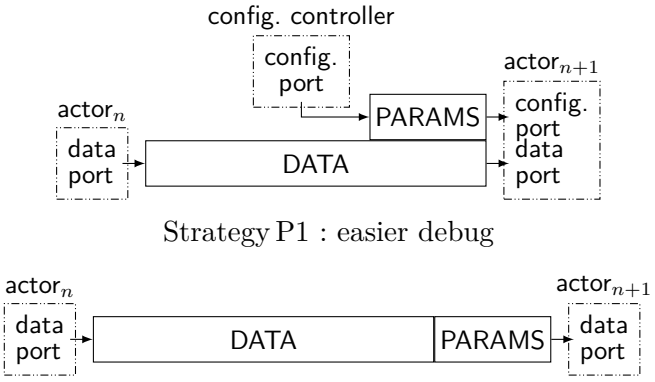


Figure 5.3 – Two propositions of strategies for the parametrization of actors

the already available SW and HW versions are archived in a formal way (e.g. using an extended version of IP-XACT such as the one proposed in [109]) since they can strongly benefit to design reuse, which is a key cost factor in Industry.

Step 4 HW selection and reconfiguration model

After the design space exploration, we can decide which actors can be implemented in HW$^{\Theta}$. After this step and for a specific architecture, we can deduce an approximate reconfiguration time for each HW configuration, as the size of the bitstream (and by extension the resource consumption) is near linearly dependant on reconfiguration time [110]. The application experts define the way the application can adapt. But the team still needs to express the way it is implemented. Fig. 5.4 presents different ways to implement the reconfiguration of an actor in HW. But first some information is mandatory to choose the best implementation. Then, the following questions, summed up in Table 5.2, must be answered for each reconfigurable actor^{A}:

Q1. Is the actor always active? If not, we can stop it to reconfigure as is done usually in systems using DRP (Strategy R1). This solution may require data buffering as detailed in [111]. If the process is always active (e.g. in some dataflow applications), we need to perform a seamless reconfiguration.

Q2. Is data parallelism applicable? (i.e. does this fall within the definition of 'single process multiple data' (SPMD)? [112]). If we can perform the computation on multiple data simultaneously (e.g. in radar and sonar systems), we can perform a reconfiguration per group of channels (Strategy R2). The quality of processing decrease but service is not

interrupted.

Q3. Is there a minimal version of this actor (a version of the algorithm which guarantees a minimum QoS)? If yes, an application expert must specify the minimal version and the HW architect have two choices: either we divide the processing in a static minimal block and a reconfigurable part (Strategy R3), or, if design space exploration showed that we can implement the process in SW, we can bypass the HW block during configuration so the computation is done in SW (Strategy R4). With Strategy R3 and Strategy R4, we can implement another accelerator in the RP using the optional I/O if the actor's computations are already done in the static part or in SW.

Finally, if none of these solutions is suitable, we can implement application reconfiguration as a reconfiguration by registers, i.e. the different configurations are implemented on the FPGA and we switch between configurations at runtime with a control bus (Strategy R5). One of these strategies must be chosen for every reconfigurable actor implemented in HW, with help of the application expert answers$^{\Theta}$. These decisions will come with a choice of interconnect types for each partition.

Our framework targets dataflow applications, thus it would not make sense to move actors implemented with Strategy R2, R3 or R4 in another actor partition, as the benefits from their reconfiguration is based on a permanent availability of a minimal version in the dataflow graph. However, actors from Strategy R1 can be implemented in the resources available from any other actor designed with Strategy R2. Moreover, they can also be implemented in the reconfigurable partition of an actor designed with Strategy R3 or R4 and configured in its minimal version. Formally, let $P_{n,m}$ be the partition designed for actor $a_{n,m}$ with n the actor id and m the implementation strategy. Any actor $a_{n,1}$ can be implemented in partitions $P_{i,1}$ if actor a_i is not implemented or in partitions $P_{i,3}$ or $P_{i,4}$ if actor a_i is configured in a minimal version.

Step 5 SW selection and reconfiguration management

After the design space exploration, we can decide which actors can be implemented in SW$^{\Sigma}$. The reconfiguration of SW processing is quite straightforward. But SW can benefit from the acceleration offered by actors implemented HW. The information of the implemented HW accelerators can be retrieved from the configuration controller and the SW architects must make sure that these accelerators are well used (write inputs, fire, read outputs). Furthermore, the SW experts must prepare the HW reconfiguration support at

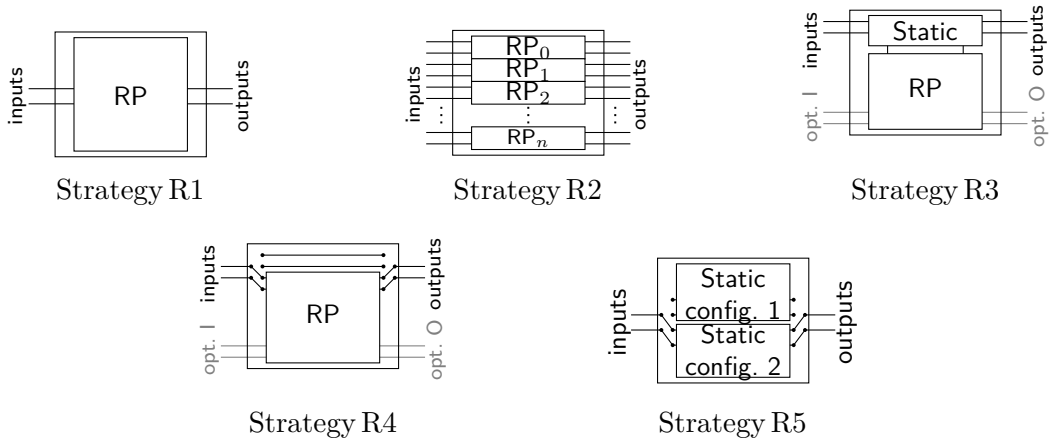


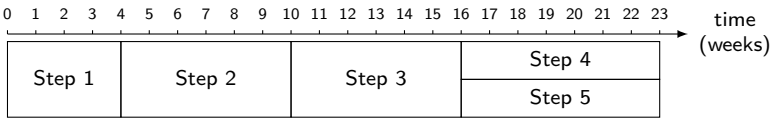
Figure 5.4 – Five propositions of strategies for reconfigurable HW blocks

SW level. For an efficient reconfiguration, the bitstreams must be prefetched from a static to a dynamic memory (e.g. SD to DDR4). Depending on the available resources and the number and size of bitstreams (either estimated or final), the SW expert can choose to prefetch either all the bitstreams or only the next possible configurations. The next possible configurations can be deduced from the configuration controller.

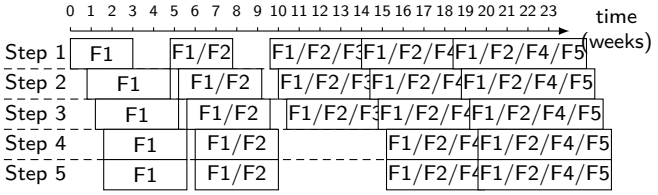
5.3.3 Agile compatibility

The clear definition of the design steps, the separation of concerns and the use of high level tools allow to modify the system easily after a first production run. We can face several modifications of the project:

1. Inclusion of new configurations: The Agile team can add configuration to existing processes to add new functionalities to the product. This can be done in the application expert's interface. The team can now follow the steps and apply minor modifications to integrate the new functionality into the product.
2. Introduction of new actors: The Agile team can add actors to enhance the system performances. This can be done in the application expert's interface. The team can now follow the steps and apply minor modifications to integrate the new processing into the product.
3. Decomposition of an actor into several actors: Based to the initial coarse-grain specification, the application experts can decide to split an actor into several actors to make one (or several) of the resulting actors reconfigurable. This is the most



(a) Waterfall



(b) Agile (Fn represent the set of desired features for iteration n)

Figure 5.5 – Adaptive system incremental design

difficult modification as this requires to perform the HW/SW exploration with the new sub-blocks.

Our framework allows all these modifications at a low extra cost.

Figure 5.5 shows an example of methodology steps, for application composed of five sets of features (F1 to F5, which represents increment of the processing flow or the inclusion of new configurations), through time with waterfall (Fig. 5.5a) and Agile (Fig. 5.5b) methodologies. With the waterfall methodology (Fig. 5.5a), all features are included in the product, which results in long and costly development steps. With the Agile methodology (Fig. 5.5b), only a subset of the functionalities are included in the product at each iteration, which results in faster development steps. The later allows early deliveries and adaptability to customer requirements.

To support the development of a complex product in faster steps, the different experts must be able to communicate and design efficiently the system. Therefore, we need to define a communication tool to ease the design of reconfigurable systems by experts with different skills and concerns.

5.4 Computer-Aided-Design tools

5.4.1 CCAD: Communication tool for agile CAD

In our framework, the agents have different views and use different tools, so they must have different interfaces to avoid information overload and guarantee separation of concerns. We decided to create an interface per speciality, so three interfaces. Figure 5.6

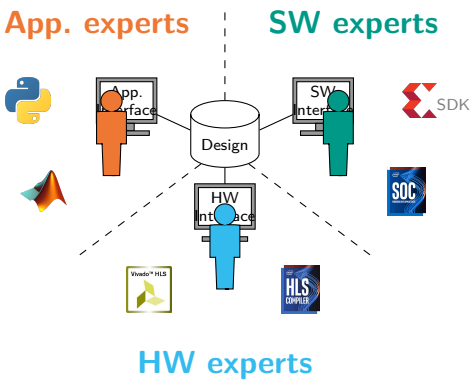


Figure 5.6 – Ensuring separation of concerns with CCAD

emphasizes how this concept can guarantee a good separation of concerns. Each expert has his/her own skills and tools and can interact with other experts through an appropriate interface.

As the system must be designed at high level and must be suitable for incremental design, we choose to use a visual definition. All agents can have access to all types of interfaces, but their spectrum of actions is limited according to their own skills.

5.4.1.1 Application interface

This interface is a drag-and-drop interface, where the architect can create actors, and connect them to form the block diagram. The agents can name the actors. The actors can be tagged as reconfigurable. The agents can indicate that a QoS indicator can be extracted from an actor, and provide a short label and a full description. A panel is associated to each actor. This panel lists the different options from Table 5.3. This panel also lists the configurations, and shows the reconfiguration controller. This controller diagram is automatically created and modified when creating new configurations, and transitions can be customized. This graphical definition will be automatically translated into a Heptagon/BZR program. When customizing controller transitions, auto-completion encourage the use of the QoS criteria previously defined. For the QoS criteria available in the actors tagged as MIB, two functions are available: *lowest(<qos label>)* and *highest(<qos label>)* which returns respectively the lowest or the highest QoS value among instances. A checklist allows the application architect to give details about the reconfigurable actor (cf. questions from Step 4).

Once the system is fully described at application level, the controller must be gener-

ated, and the behaviour of the adaptive system can be observed. Application experts can eliminate configurations by adding rules all along the design process, such as: *configuration C1 and C5 can not be implemented simultaneously.* These rules are expressed with the syntax of Heptagon/BZR contracts[113]. The application experts must validate the remaining configurations and provide test files in the interface.

5.4.1.2 HW interface

The interface is a graphical view generated from the application block diagram. It features static processes and dynamic processes. A panel is associated with each actor and lists the options from Table 5.3 filed by the application experts and the configurations if the actor is reconfigurable. These information are in read-only mode from the HW experts perspective. The HW experts can tag the actors to tell that they can be implemented in HW. They can fill resource consumption and execution time information in the properties list while doing design space exploration with Vivado HLS of Intel HLS compiler. When a HW actor is reconfigurable, they must specify the reconfiguration strategy (Fig. 5.4) and the reconfiguration time (estimated at Step 3 and measured at Step 4).

The HW experts can now simulate the system with post-codesign information (resources usage, latency) to see which configurations can be reached and eliminate the configurations which do not seem implementable because of resources consumption with Heptagon/BZR contracts. They must validate the HW actors for remaining configurations using the test files provided by application experts. They must provide their test files as they may differ from the application ones (e.g. because of the use of fixed-point computations).

5.4.1.3 SW interface

The interface is a graphical view generated from the application block diagram. It features static processes and dynamic processes. A panel is associated with each actor and lists the options from Table 5.3 filed by the application experts and the configurations if the actor is reconfigurable. These information are in read-only mode from the SW expert perspective. They can fill execution time information in the properties list while doing design space exploration by running cross-compiled code on target processor using Xilinx SDK or Intel SoC FPGA EDS.

The SW experts can now simulate the system with post-codesign information (latency) to see which configurations can be reached and eliminate the configurations which do

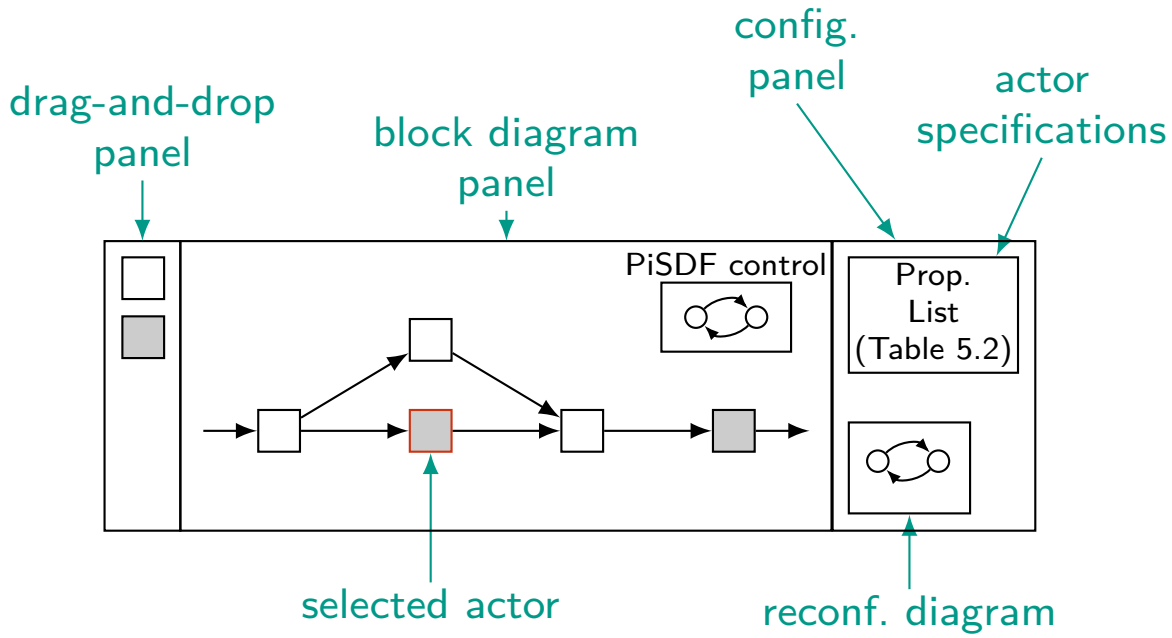


Figure 5.7 – Application GUI

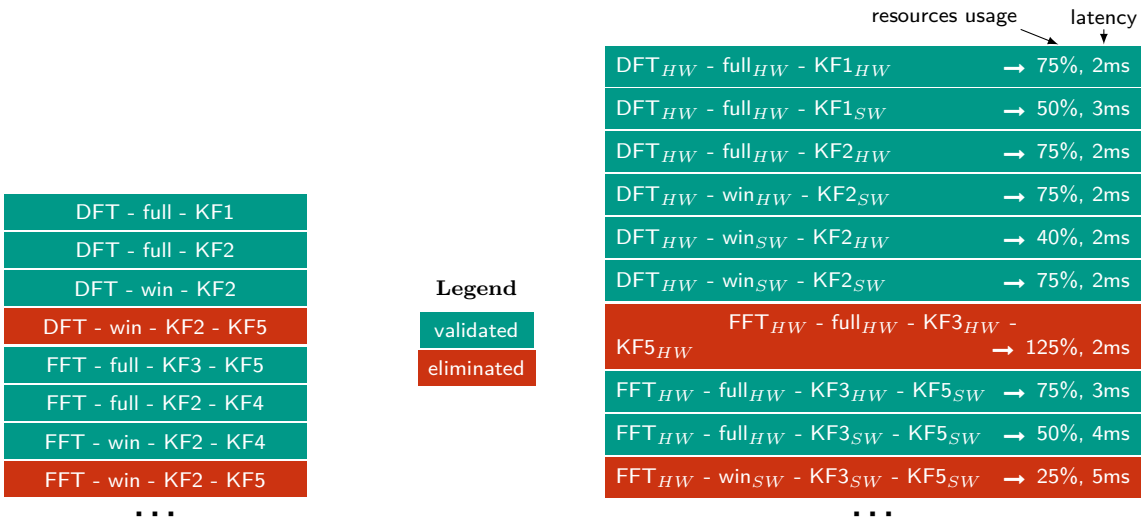
not respect latency constraints with Heptagon/BZR contracts (explained in 5.4.2). They must validate the SW actors for remaining configurations using the test files provided by application experts. They must provide their test files as they may differ from the application ones.

5.4.1.4 Inter-agent communications

The first communication medium is the development environment. Through the interfaces, the agents modify the definition of the system and every agent can observe the current state of the system description. The second communication medium is a messaging interface in the CCAD tool where agents can send queries related to specific points or system design choices. The agents specialized in the domain of the questioned design choice can answer the queries.

5.4.2 Automatically generated reconfiguration control

The different experts created a controller for every configurations and partitions in the GUI. These controllers need to be generated, tested and implemented. To do so, we use Heptagon/BZR as a backend. The Heptagon/BZR tools allow to generate a correct-by-construction global controller from several small controllers. Our methodology would



(a) Applicative level (Step 2) (b) Implementation level (Step 3)

Figure 5.8 – Lists of configurations. Simplified and non-exhaustive list from the example of Section 5.5.

benefit from a way to list configurations to allow experts to find flaws in their original specifications. However, displaying an exhaustive list of reachable configurations make no sense as the configuration space grows exponentially w.r.t. the number of states. The representation of the configuration space to human operators is an active research field. For our methodology we use the interim solution which to simulate the generated controller and observe the reached configurations, so that the experts can eliminate some combinations based on application or performances aspects. This action can be performed after Step 2 and Step 3. Fig. 5.8a shows a snippet of the configurations listing of the case study from Section 5.5 at application level. It shows that the configurations must be either eliminated or validated after Step 2. Fig. 5.8b shows the same operation after Step 3.

5.5 Case study

5.5.1 Agents team specification

To evaluate our framework, we develop an architecture combining our previous work and extended with reconfiguration of the detection process. We will present the development of the radar chain following our design rules. Our case study features three agents:

Agent 1. HW architect/developer, SW architect/developer, application developer

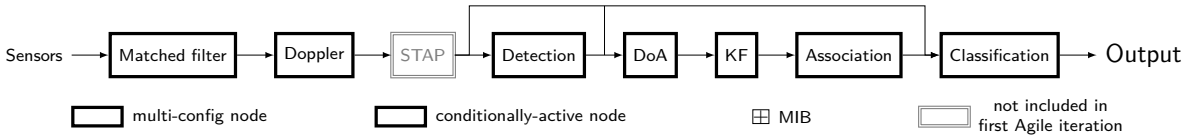


Figure 5.9 – Static application description

Agent 2. Application architect/developer, SW developer

Agent 3. HW developer, SW developer

Table 5.1b presents these specifications. With these agents, we followed the methodology steps to develop an adaptive radar architecture. Section 5.5.2 describes the development step-by-step.

5.5.2 Adaptive radar system definition

Step 1 Static application definition

Fig. 5.9 presents the description of the static radar processing system defined by Agent 2 (actors with double gray lines are not implemented in this first implementation). We can see that the actors describe classic elements of a radar tracking system. First, a matched filter filters sensors data in the range axis. Then, a Doppler extraction is performed by Fourier transform. CA-CFAR detection is performed on the signal and the detected plots are subject to a direction of arrival estimation. The trajectory of detected objects are filtered by Kalman filters. An Association filter create and delete tracks and associate plots on a track based on the Kalman filters innovation likelihood. Eventually, a classification tree select labels to identify the tracked objects. Agent 2 specifies that the maximum latency should be 4ms to ensure that the radar system is responsive enough.

Step 2 Application reconfiguration

Agent 2 knows that there are several ways to perform the Doppler processing. He defines this actor as reconfigurable, and lists two configurations (inspired from [3]). Fig. 5.10 shows an example of the definition of the *Doppler* actor for the two configurations. The *slowt* static parameter is used to modify the radar repetition frequency, and can be modified between two graphs iterations. The value of *slowt* is used by several actors in our example and will be modified at runtime by a PiSDF controller. Therefore, our actors implementations are compatible with *slowt* changes. Agent 2 also knows that an object

```
actor Doppler:
  inputs:
    fp<16, 2> inChannels[4]
  outputs:
    fp<16, 2> outChannels[4]
  sparameters:
    slowt ∈ {32, 64, 128}
  cparameters:
    N.A.
  tokens:
    inChannels => 125 × 128
    outChannels => 125 × 128
  description:
    Compute FFT in the slow time axis.
    Perform a FFT of size slowt.
end
```

(a) FFT mode PiSDF description

```
actor Doppler:
  inputs:
    fp<16, 2> inChannels[4]
    fp<16, 2> inCoefs
  outputs:
    fp<16, 2> outChannels[4]
  sparameters:
    slowt ∈ {32, 64, 128}
    rshift ∈ {0, ..., 125 − 32}
    dshift ∈ {0, ..., 128 − 32}
  cparameters:
    N.A.
  tokens:
    inChannels => 125 × 128
    outChannels => 32 × 32
    inCoefs => slowt × 32
  description:
    Compute DFT in the slow time axis.
    DFT input size is slowt.
    Keep 32 of 128 spectral samples
    (shifted by dshift)
    for 32 of 125 ranges samples
    (shifted by rshift).
end
```

(b) DFT mode PiSDF description

Figure 5.10 – PiSDF description of Doppler processing. Description details are omitted for clarity.

can be tracked by different Kalman filter (KF) models, and that we can change the model depending on the object dynamic. He identifies five interesting models. Agent 1, who has application knowledge, notes that we can reduce the detection window when the object is already detected to reduce execution time. He defines two configurations for the detection actor: *full* for a detection on the whole radar image, and *win* for a windowed detection. Agent 2 tags the direction-of-arrival (DoA), KF, association and classification actors as conditionally active, because they don't have to run if no target is detected. Agent 2 now describes the reconfigurable aspect of the application. To take reconfiguration decision, he must propose QoS criteria. He knows that the validation score of the association process can be used to confirm detection [10] and adds it as a QoS criterion. He knows that the innovation likelihood of the Kalman filter can be used to estimate the model-measurement

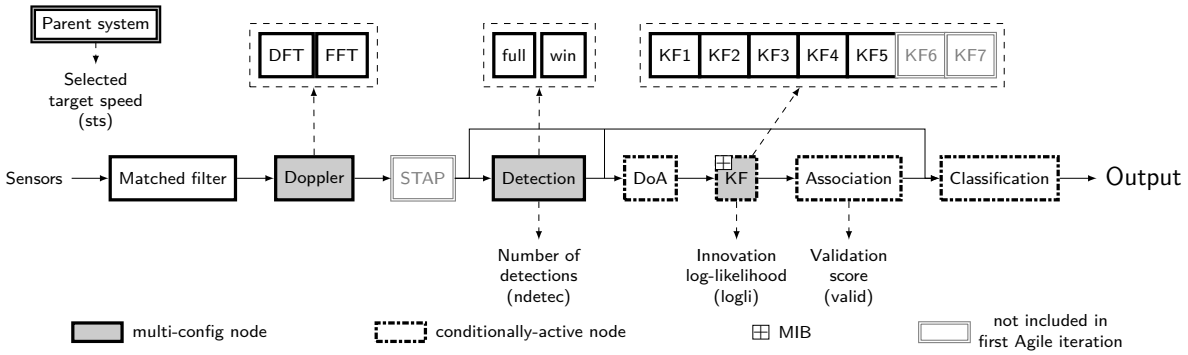


Figure 5.11 – Dynamic application description

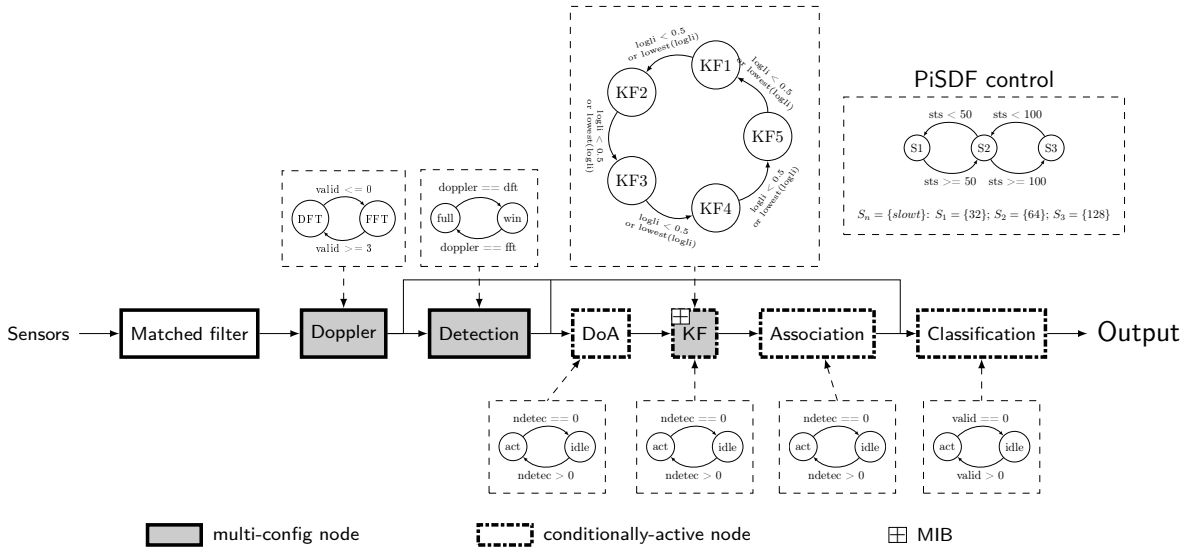


Figure 5.12 – Dynamic application reconfiguration control description

adequacy, which is a Kalman filter QoS indicator [9], and adds it as another QoS criterion. Eventually, he adds the number of detection as a QoS criterion, because some process will only be active if detection is successful. Fig. 5.11 presents the description of available configurations, conditionally active actors, MIB and available QoS criteria.

Fig. 5.12 illustrates the reconfiguration control designed by the application architect. The reconfiguration of the *Doppler* actor should occur when an object of interest detection is confirmed. This information is obtained by comparing the validation score to a threshold. In Fig. 5.12, the reconfiguration diagram over the Doppler block, defined by Agent 2, implements this strategy. Fig. 5.12 also shows the control designed by the application architect to describe the conditions in which the conditionally active processes goes idle or active. Agent 2 list the configurations and prune the configurations where

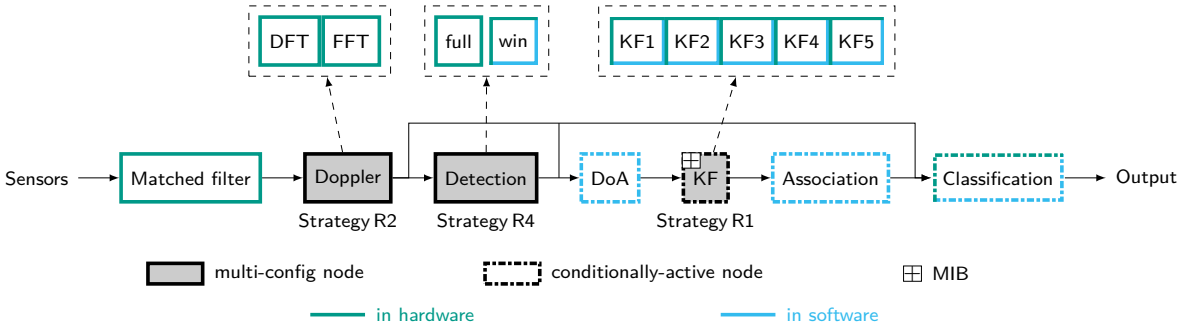


Figure 5.13 – HW/SW first partitioning

KF2 and KF5 are implemented together because the models are too close from each other (Fig. 5.8a). Agent 2 validates the remaining configurations with Python3 and stores test vectors to allow implementation verification.

Step 3 Design space exploration

Agent 1 and 3 agree to implement the dataflow parametrization with Strategy P1 to be able to easily debug the application as this is the first development iteration. This design choice may change in a later development iteration. Fig. 5.13 illustrates the observations made by the HW and SW experts during codesign. Agents 1 and 3 tag the actors which they are sure can be implemented in HW and SW. The answer is really straightforward for some actors. *Matched filter* and *Doppler* are massively parallel process which must be executed for every radar channels (SPMD on 16 channels in our example), so they are tagged as HW blocks. For the other actors, design exploration is performed. Agents 1 and 3 carry out fast prototyping of the actors, with Vivado HLS for the HW and on the ARM Cortex A53 for the SW. They report execution times and resource consumption in their respective interface (see Table 5.4 for resource consumption and latency on a Zynq Ultrascale+ XCZU9EG target). Agent 1 simulate the controller behaviour and prune configurations where the resource consumption or the latency is too high (Fig. 5.8b). He then validates the remaining configurations by using the test vectors provided by Agent 2. If the output results differ from the original test vectors, e.g. because of the use of fixed-point, they must be subject to approval by Agent 2.

Table 5.3 – Filled specifications for reconfigurable actors

Question	Doppler	Detection	KF
Reconfigurable	✔	✔	✔
Always active	✔	✔	✖
MIB	✖	✖	✔
Configurations	DFT, FFT	full, win	1, 2, ..., 5
QoS criteria	—	# of detections	Innov. likelihood
Actor description	Fig. 5.10b, Fig. 5.10a	omitted	omitted
HW implementable	✔, ✔	✔, ✔	✔, ✔, ..., ✔
SW implementable	✖, ✖	✖, ✔	✔, ✔, ..., ✔
Data parallelism	✔	✖	✖
Minimal version exists	✖	✔	✖
Minimal version	—	win	—
Implem. strategy	Strategy R2	Strategy R4	Strategy R4

Required at step:
Step 3 Step 4/Step 5 Step 4

Step 4 HW selection and reconfiguration model

The *full Detection* actor is selected for a HW implementation because the execution is way faster than the SW implementation. The *win Detection* actor could be implemented in SW or HW because the computation is lighter than the *full Detection* mode. The *Kalman* actor could be implemented in SW or HW, and as we need a maximum number of Kalman filters to run simultaneously, we choose to implement Kalman filters in HW and SW.

Agent 2 answers the three questions from Section 5.3.2 Step 4 about the reconfigurable actors. Table 5.3 presents the answers to the questions for the three reconfigurable actors. From these information, agent 1 chooses to implement the *Doppler* actor with channel reconfiguration (Strategy R2). Agent 1 chooses to implement the *Detection* actor with a a data redirection to SW and a RP (Strategy R4). With this strategy, we can free HW to implement an additional Kalman filter when detection is transferred to SW.

Step 5 SW selection and reconfiguration management

Agents 1 and 3 conclude from Step 3 that the *DoA* and *Association* actors should be implemented in SW. The *Classification* actor could be implemented in both HW and SW, but the chosen algorithm is light so agents 1 and 3 decide to implement it as SW. This could be changed in a later iteration on the project (e.g. a new story imposes that a new category of objects should be recognized, and therefore acceleration is mandatory).

The bitstreams generated at Step 4 take a total size of about 100Mb (21Mb for DFT/FFT + $\sim$ 84Mb for detec + KF) while the available DDR memory size is 2Gb.

Agent 1 decides to prefetch all the bitstreams in DDR as the application is not memory bound.

5.5.3 Expected gains

5.5.3.1 Architecture gains

Table 5.4 presents the resource consumption and the latency of each actors and configurations, obtained with Vivado HLS and on a Zynq Ultrascale+ XCZU9EG. The execution time is reduced by 2.7ms when the system switches from FFT to DFT, thus freeing time for other tasks. When the system is in DFT mode, the detection is transferred to SW and the RP can be exploited by a Kalman filter to speed up further the application. These two optimizations save a lot of computing time while switching from to a radar searching to tracking mode which means that we can implement more and/or better performance processes on the platform. This can be done in a future project iteration (e.g. in Section 5.5.3.2).

The saved resources, compared to an implementation where all the HW configurations are implemented, are at least equal to the minimum of each resources used by accelerators which share partitions:

- DFT/FFT : 4889 LUT, 2648 FF, 24 DSP, 48 BRAM
- Detec./KF1-5 : 16964 LUT, 20881 FF, 82 DSP, 36 BRAM

These optimizations freed a lot of resources, thus allowing us to implement the same system on a cheaper system or to implement more accelerators in HW in a future development iteration.

5.5.3.2 Future functionalities

After this first development iteration, it is possible to improve the system by adding new functionalities. The actors with double gray lines in Fig. 5.9 and 5.11 shows an example of the incrementation of the previous solution. We can see that two Kalman filter models were added to the *KF* actor, and that one actors was added to the processing chain. Following the different methodology steps and reusing previous definitions and costs estimations for the implementation of existing functions, the incrementation is possible at a lower cost than if we had to redesign the whole system.

Table 5.4 – Resource consumption and latency of actors from Fig. 5.13

Actor	Config.	LUT	FF	DSP	BRAM	latency(ms)	
						HW	SW
Match. (×4)	—	25020	28290	50	22	0.14	—
Doppler (×4)	DFT	4889	2648	43	48	0.2	—
	FFT	6697	9136	24	6	2.9	—
Detection	full	16964	20881	82	42	0.09	—
	win	—	—	—	—	—	0.6
DoA	—	—	—	—	—	—	2.5
KF	KF1	32688	29582	154	30	0.07	0.3
	KF2	41398	40674	174	34	0.12	0.4
	KF3	41398	40674	174	34	0.12	0.4
	KF4	43095	34562	186	165	0.17	0.6
	KF5	42337	42750	202	36	0.12	0.4
Assoc.	—	—	—	—	—	—	~ 0
Classif.	—	—	—	—	—	—	~ 0

5.6 Conclusion and future work

We propose CANDID, a methodology for the high level specification of reconfigurable SoC designs. This methodology allows to leverage application knowledge to make application specific optimizations by adapting the resources use at runtime. CAD tools for the support of this methodology are described. Eventually, a case study describe the virtual development of a realistic radar tracking system.

Although the case study does not prove the efficiency of our methodology for every application, it illustrates how this framework could be used in an industrial project. All of our framework guidelines are justified. A future work is to apply this methodology in an industrial development team, to confirm the framework relevance and to propose adjustments when necessary. This future work could also offer new innovative adaptive architectures. Another future work would be to consider hierarchical reconfiguration in the methodology. For example in radar systems, we can use either classic radar tracking or synthetic aperture radar. We could reconfigure the system to switch between these modes, but we can also modify some sub-processes. Eventually, the CANDID methodology will be integrated as a plugin in the Capella model-based system engineering tool[114] to ease the adoption of this methodology in Industry.

CONCLUSION AND FUTURE WORK

Conclusion

The arise of multi-transceivers antennas in air- and UAV-borne systems has allowed the use of many modes and efficient algorithms. However, this has led to the complication and enlargement the RF front end. To reduce the size of these systems and enhance radar integration, many functions were transfered into the digital part of the system. But scanning several antennas at high rates produces huge amounts of data, that must be processed by the on-board computing resources. In addition, the several modes available on radar systems require different processing, implemented in both HW and SW. This continuous increase of data and modes available on embedded radar systems puts pressure on the processing systems that must adapt to meet the challenge. To relax the pressure on these systems, the improvement rate of computing hardware, such as CPU and FPGA, is not sufficient. However, it is possible to enhance systems by reallocating computational components at runtime. This can be performed by HW or SW dynamic reconfiguration.

In the first chapter, we introduced key concepts of radar tracking processing and computational systems, necessary for understanding the rest of the thesis. We first detailed the processing blocks commonly implemented in tracking radars. Then, we explained the concept of reconfigurable HW and SW systems. Finally, we demonstrated the need for reconfigurable systems and criteria, which can enable highly integrated radars to handle more modes and achieve higher performance than static systems.

In the second chapter, we presented a state of the art divided in three sections. Firstly, we introduced the dynamic partial reconfiguration of computational systems based on FPGA. This technology allows optimizations and self repairing of computing systems, but sometimes the best configuration depends on the mission context and quality of service criteria must be used to select the best configuration at runtime. We observed that literature is not extensive on radar systems dynamic partial reconfiguration based on application criteria. Secondly, we presented the three algorithms used as case studies in this thesis. Even though these algorithms have been widely studied, the application of the adaptivity concept permits new optimizations. Thirdly, we presented the reconfiguration

methodologies used for the design of HW/SW reconfigurable systems. Although methods were proposed for the design of reconfigurable systems, these works often require deep technical knowledges. Therefore, they neglect application experts who often have limited knowledge of the underlying implementation target. Finally, we conclude this chapter by positioning the thesis work in the literature.

We divided the third chapter in two sections which present architecture case studies based on QoS criteria. The first section presents a tracking system which use seven different models to track virtual targets. Only two filters can be implemented at the same time and the system does not know *apriori* which model is the most adapted. The quality of tracking is analyzed to know which model is most adapted in order to reconfigure the other accordingly. We showed that the use of this criterion can drive the system reconfiguration with success and we demonstrated that their reconfiguration can improve tracking. The second section presents a reconfigurable architecture for a sixteen-sensor radar which switches between two well-known algorithms, the discrete Fourier transform sum and the fast Fourier transform, to improve the latency of a radar system during tracking. This switch is controlled by a criterion which confirms the correct detection of a target. We implemented and tested this system in a hardware-in-the-loop simulation that showed the principle of switching between DFT and FFT and confirmed the expected gains on both latency and FPGA resources consumption, thus demonstrating the relevance of our approach.

In the fourth chapter, we presented an original method for the adaptation of space-time adaptive processing. This method, based on machine learning, predicts the most adapted filtering dimensions at runtime, in order to reduce computing complexity while maximizing the quality of results. We tested this method on a dataset created from various interferences models. This preliminary work shows encouraging results for predicting optimal filtering dimensions using machine learning. The resulting algorithm is reconfigured at runtime in software.

In the fifth chapter, we presented a methodology for efficient design of reconfigurable HW/SW processing systems, with existing tools. To facilitate the exchanges between heterogeneous experts, we specified the actors, their respective roles and we defined design patterns. A case study involving the development of a reconfigurable system based on the architectures presented in Chapter 3 illustrates this methodology. We believe that this high abstraction level methodology can help to design reconfigurable systems with the current tools and independently from the used tools (e.g. Xilinx or Intel frameworks).

In this work, we explored and demonstrated improvements which can be obtained by the reconfiguration of embedded radar systems. It showed that for a state-of-the-art computing platform, we can only reach maximum performances by reallocating the computing resources to the algorithms which are the most adapted to the current mission context.

Limitations

This section presents the limits of the solutions proposed in this document. The first limitation is the limited availability of real data to conduct our tests. Indeed, since Thales is specialized in defense, we could not use real data to test our systems, so we used synthetic data. We also observe specific limitations in the different works:

1. For the Kalman filter reconfiguration, our models are simple and elaborated for proof of concept. To obtain similar results in real-life tracking, we would need real targets measurements as well as various models specific to the different targets types. In addition, the next models after reconfiguration are selected in a round-robin fashion, whereas the next model could be selected in a more intelligent way.
2. For the reconfigurable Doppler processing, our architecture was tested only for a subset of parameters, such as range-Doppler map size and PRF, so the achieved gains in performances and saved hardware logic are not guarantied for every architecture. In real-life application this gain could be either higher or lower.
3. For the reconfigurable Space-Time Adaptive Processing, the optimal dimensions selection is based on empiric rules, hence it is not strictly optimal. In addition, the method used for dimensions reduction is suboptimal. Indeed, event though this method reduces the complexity and the condition number, it drops samples and hence reduces the SINR as well as Doppler and angle resolution. Even considering this drawback, this study provides a proof of concept that needs to be improved before it can be used in real systems.
4. The HW/SW reconfigurable embedded systems design methodology has interesting concepts, but was not tested by a development team to confirm the relevance of the method. In addition, we described a software interface to support our methodology but we did not have the time to develop it. However, the methodology can be used without a software interface and with current development tools.

Like many studies, ours has several limitations. However, all of these limitations can be pushed back, and much future work can improve the results already obtained in this thesis.

Future work

This section presents the most relevant perspectives that arise from our work. The reconfiguration was explored in several fields, but the reconfiguration of embedded radar systems has received little attention in the literature. This thesis covers radar problems ranging from signal processing applications to hardware implementations. The result is a wide scope of application that has left us with many avenues to explore, the most important of which are:

1. Firstly, based on the identified criteria and with the help of the CANDID methodology, we could conduct a systematic study of hardware/software reconfiguration for all the main radar function.
2. Secondly, we can foresee future work specific to each case study:
   (a) For the reconfigurable Kalman filter, but we could further enhance performances by combining our architecture with the principle of interacting multiple model [50]. We would get a system which not only select the most suitable models, but also combine the prediction of available models to improve tracking. In addition, the current model selection at reconfiguration time is based on round-robin scheduling, but we could create a scheduling based on similarity between models. Indeed, if a model is not adapted, it is unlikely that a similar model will perform well. A more intelligent model selection would be a step forward from QoS-based to QoS-driven reconfiguration.
   (b) For the reconfigurable Doppler processing, our architecture is rigid. The PRF and the reduced DFT window dimensions are fixed. We could include this architecture into a real radar system which tracks multiple target and adapts its PRF to observe and solve the inherent problems of these adaptations.
   (c) For the Space-Time Adaptive Processing, the proposed approach is based on machine-learning. The performance depends on the database, and in our case the database outputs (the optimal filtering dimensions) depend on empiric

rules. We could adapt our approach to select them through performances criteria, i.e. the improvement factor, the computing complexity and the condition number. In addition, we could find a simple algorithm method to filter out the cases where there are no interferences, since these case are where the ratio of prediction and computing complexity is the higher, hence the cases where our algorithm is the less efficient. This could be achieved with a method based on the kurtosis computation [115], which gives results close to zeros on Gaussian noises. Finally, a hardware architecture could be created to fully benefit from our algorithm.

3. Finally, the proposed methodology for the design of reconfigurable HW/SW systems could be applied to a full-scale project to confirm its relevance, find imperfections to improve it. In addition, we could create a tool to support this methodology, possibly as an extension to current design tools such as Capella.

This thesis focused on the optimization of embedded radar applications through reconfiguration, and we did not consider fault detection and correction through reconfiguration. This is a critical problem for maximizing processing quality. For example, multiple sensors antennas may encounter a transceiver failure. Since processing is performed for multiple channels in parallel, it would be interesting to detect transceiver failures and reconfigure the processing resources dedicated to that transceiver's channel to improve the processing performed on the other channels.

Published work

Two papers were published during this thesis:

1. Section 3.1 was published under the following reference:
   J. Mazuet, I.-h. Atchadam, D. Heller, *et al.*, « QoS Driven Dynamic Partial Reconfiguration: Tracking Case Study », *in 14th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC 2019)*, York, United Kingdom, Jul. 2019.
2. Section 3.2 was published under the following reference:
   J. Mazuet, M. Narozny, C. Dezan, *et al.*, « A Seamless DFT/FFT Self-Adaptive Architecture for Embedded Radar Applications », *in The International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, Aug. 2020.

Two additional papers, based on the content of Chapters 4 and 5, will be submitted to journals.

BIBLIOGRAPHY

[1] F. J. Pollack, « New microarchitecture challenges in the coming generations of CMOS process technologies (keynote address) », *in Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture*, 1999, p. 2.

[2] J. Mazuet, I.-h. Atchadam, D. Heller, C. Dezan, M. Narozny, and J.-P. Diguet, « QoS Driven Dynamic Partial Reconfiguration: Tracking Case Study », *in 14th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC 2019)*, York, United Kingdom, Jul. 2019.

[3] J. Mazuet, M. Narozny, C. Dezan, and J.-P. Diguet, « A Seamless DFT/FFT Self-Adaptive Architecture for Embedded Radar Applications », *in The International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, Aug. 2020.

[4] W. Melvin, « A STAP overview », *IEEE A&E systems magazine*, vol. 19, pp. 19–34, 2004.

[5] S. Blake, « OS-CFAR Theory for Multiple Targets and Nonuniform Clutter », en, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, *6*, pp. 785–790, Nov./1988, ISSN: 00189251. DOI: 10.1109/7.18645.

[6] R. Schmidt, « Multiple Emitter Location and Signal Parameter Estimation », en, *IEEE Transactions on Antennas and Propagation*, vol. 34, *3*, pp. 276–280, Mar. 1986, ISSN: 0096-1973. DOI: 10.1109/TAP.1986.1143830.

[7] R. Roy and T. Kailath, « ESPRIT-Estimation of Signal Parameters via Rotational Invariance Techniques », en, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, *7*, pp. 984–995, Jul. 1989, ISSN: 00963518. DOI: 10.1109/29.32276.

[8] C. M. S. See and A. B. Gershman, « Direction-of-arrival estimation in partly calibrated subarray-based sensor arrays », *IEEE Transactions on Signal Processing*, vol. 52, *2*, pp. 329–338, 2004.

[9] P. Tandeo, P. Ailliot, M. Bocquet, A. Carrassi, T. Miyoshi, M. Pulido, and Y. Zhen, « Joint Estimation of Model and Observation Error Covariance Matrices in Data Assimilation: A Review », en, *arXiv:1807.11221 [stat]*, Jul. 2018. arXiv: 1807.11221 [stat].

[10] Y. Bar-Shalom and F. Daum, « The Probabilistic Data Association Filter », en, *IEEE Control Systems*, vol. 29, *6*, pp. 82–100, Dec. 2009, ISSN: 1066-033X, 1941-000X. DOI: 10.1109/MCS.2009.934469.

[11] M. Andric, Z. Durovic, and B. Zrnic, « Ground Surveillance Radar Target Classification Based On Fuzzy Logic Approach », en, *in EUROCON 2005 - The International Conference on "Computer as a Tool"*, Belgrade, Serbia and Montenegro: IEEE, 2005, pp. 1390–1392, ISBN: 978-1-4244-0049-2. DOI: 10.1109/EURCON.2005.1630220.

[12] G. Kouemou, C. Neumann, and F. Opitz, « Exploitation of Track Accuracy Information in Fusion Technologies for Radar Target Classification Using Dempster-Shafer Rules », en, p. 7, Jul. 2009.

[13] N. M. Nasrabadi, « Deeptarget: An automatic target recognition using deep convolutional neural networks », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, *6*, pp. 2687–2697, 2019.

[14] K. El-Darymli, E. W. Gill, P. McGuire, D. Power, and C. Moloney, « Automatic target recognition in synthetic aperture radar imagery: A state-of-the-art review », *IEEE access*, vol. 4, pp. 6014–6058, 2016.

[15] U. K. Majumder, E. P. Blasch, and D. A. Garren, *Deep Learning for Radar and Communications Automatic Target Recognition*. Artech House, 2020.

[16] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, « A Survey on Engineering Approaches for Self-Adaptive Systems », en, *Pervasive and Mobile Computing*, vol. 17, pp. 184–206, Feb. 2015, ISSN: 15741192. DOI: 10.1016/j.pmcj.2014.09.009.

[17] D. K. Ho, « QoS-aware resource and energy management for autonomous mobile », PhD thesis, Université Côte d'Azur, 2020.

[18] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl Göschka, « On Patterns for Decentralized Control in Self-Adaptive Systems », en, *in Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24 - 29, 2010 ; Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science 7475, R. de Lemos and I. S.o.S.E.f.S.-A.S. D. 2010.10.24-29 Wadern) : Eds., Berlin: Springer, 2013, pp. 76–107, ISBN: 978-3-642-35812-8 978-3-642-35813-5.

[19] Intel FPGA, *AN 806: Hierarchical Partial Reconfiguration Tutorial: for Intel Arria 10 GX FPGA Development Board*, en, 2021.

[20] Xilinx, *UG 1137: zynq UltraScale+ MPSoC: software developers guide*, 2020.

[21] I. FPGA, *AN 798: Partial Reconfiguration with the Arria 10 HPS*, 2017.

[22] E. Seguin, R. Tessier, E. Knapp, and R. W. Jackson, « A Dynamically-Reconfigurable Phased Array Radar Processing System », en, *in 2011 21st International Conference on Field Programmable Logic and Applications*, Chania, Greece: IEEE, Sep. 2011, pp. 258–263, ISBN: 978-1-4577-1484-9. DOI: 10.1109/FPL.2011.52.

[23] Y. Zhang, Z. Wang, and J. Wang, « Integrated Radar Signal Processing Using FPGA Dynamic Reconfiguration », en, *in 2016 CIE International Conference on Radar (RADAR)*, Guangzhou, China: IEEE, Oct. 2016, pp. 1–4, ISBN: 978-1-5090-4828-1. DOI: 10.1109/RADAR.2016.8059575.

[24] E. Chamouard, « Radars aéroportés multifonctions (in french with english abstract) », fr, p. 24, 2013.

[25] M. A. Richards, J. A. Scheer, and W. A. Holm, Eds., *Principles of Modern Radar. Vol.1: Basic Principles*, en, first published 2010, reprinted with corrections 2015. Raleigh, NC: SciTech Publ, 2015, ISBN: 978-1-891121-52-4.

[26] D. A. James, *Radar Homing Guidance for Tactical Missiles*, en. London: Macmillan Education UK, 1986, ISBN: 978-1-349-08604-7 978-1-349-08602-3. DOI: 10.1007/978-1-349-08602-3.

[27] C. See and A. Gershman, « Direction-of-Arrival Estimation in Partly Calibrated Subarray-Based Sensor Arrays », en, *IEEE Transactions on Signal Processing*, vol. 52, *2*, pp. 329–338, Feb. 2004, ISSN: 1053-587X. DOI: 10.1109/TSP.2003.821101.

[28] P. M. McCormick, S. D. Blunt, and J. G. Metcalf, « Simultaneous radar and communications emissions from a common aperture, part I: Theory », *in 2017 IEEE Radar Conference (RadarConf)*, IEEE, 2017, pp. 1685–1690.

[29] P. M. McCormick, B. Ravenscroft, S. D. Blunt, A. J. Duly, and J. G. Metcalf, « Simultaneous radar and communication emissions from a common aperture, part II: experimentation », *in 2017 IEEE Radar Conference (RadarConf)*, IEEE, 2017, pp. 1697–1702.

[30] R. Tessier, K. Pocek, and A. DeHon, « Reconfigurable Computing Architectures », en, *Proceedings of the IEEE*, vol. 103, *3*, pp. 332–354, Mar. 2015, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2014.2386883.

[31] K. Vipin and S. A. Fahmy, « FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications », en, *ACM Computing Surveys*, vol. 51, *4*, pp. 1–39, Sep. 2018, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3193827.

[32] F. Fons, M. Fons, E. Cantó, and M. López, « Real-Time Embedded Systems Powered by FPGA Dynamic Partial Self-Reconfiguration: A Case Study Oriented to Biometric Recognition Applications », en, *Journal of Real-Time Image Processing*, vol. 8, *3*, pp. 229–251, Sep. 2013, ISSN: 1861-8200, 1861-8219. DOI: 10.1007/s11554-010-0186-1.

[33] C. Steiger, H. Walder, and M. Platzner, « Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks », *IEEE Transactions on Computers*, vol. 53, *11*, pp. 1393–1407, 2004. DOI: 10.1109/TC.2004.99.

[34] B. Pratt, M. Caffrey, J. F. Carroll, P. Graham, K. Morgan, and M. Wirthlin, « Fine-Grain SEU Mitigation for FPGAs Using Partial TMR », *IEEE Transactions on Nuclear Science*, vol. 55, *4*, pp. 2274–2280, 2008. DOI: 10.1109/TNS.2008.2000852.

[35] F. Siegle, T. Vladimirova, O. Emam, and J. Ilstad, « Adaptive FDIR framework for payload data processing systems using reconfigurable FPGAs », *in 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, 2013, pp. 15–22. DOI: 10.1109/AHS.2013.6604221.

[36] R. Dorrance, A. Belogolovy, H. Wang, and X. Zhang, « A Digital Root Based Modular Reduction Technique for Power Efficient, Fault Tolerance in FPGAs », *in*

2020 30th International Conference on Field-Programmable Logic and Applications (FPL), 2020, pp. 341–346. DOI: 10.1109/FPL50879.2020.00063.

[37] N. Montealegre, D. Merodio, A. Fernandez, and P. Armbruster, « In-Flight Reconfigurable FPGA-Based Space Systems », en, *in 2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Montreal, QC: IEEE, Jun. 2015, pp. 1–8, ISBN: 978-1-4673-7501-6. DOI: 10.1109/AHS.2015.7231177.

[38] J.-P. Diguet, Y. Eustache, and G. Gogniat, « Closed-Loop-Based Self-Adaptive Hardware/Software-Embedded Systems: Design Methodology and Smart Cam Case Study », en, *ACM Transactions on Embedded Computing Systems*, vol. 10, *3*, pp. 1–28, Apr. 2011, ISSN: 15399087. DOI: 10.1145/1952522.1952531.

[39] C. Hireche, C. Dezan, S. Mocanu, D. Heller, and J.-P. Diguet, « Context/Resource-Aware Mission Planning Based on BNs and Concurrent MDPs for Autonomous UAVs », en, *Sensors*, vol. 18, *12*, p. 4266, Dec. 2018, ISSN: 1424-8220. DOI: 10.3390/s18124266.

[40] Y. Eustache and J.-P. Diguet, « Specification and OS-Based Implementation of Self-Adaptive, Hardware/Software Embedded Systems », en, *in Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis - CODES/ISSS '08*, Atlanta, GA, USA: ACM Press, 2008, p. 67, ISBN: 978-1-60558-470-6. DOI: 10.1145/1450135.1450151.

[41] X. Zhang, X. Shao, G. Provelengios, N. K. Dumpala, L. Gao, and R. Tessier, « Scalable Network Function Virtualization for Heterogeneous Middleboxes », en, *in 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa, CA, USA: IEEE, Apr. 2017, pp. 219–226, ISBN: 978-1-5386-4037-1. DOI: 10.1109/FCCM.2017.24.

[42] F. Sironi, M. Triverio, H. Hoffmann, M. Maggio, and M. Santambrogio, « Self-Aware Adaptation in FPGA-Based Systems », en, *in 2010 International Conference on Field Programmable Logic and Applications*, Milan, Italy: IEEE, Aug. 2010, pp. 187–192, ISBN: 978-1-4244-7842-2. DOI: 10.1109/FPL.2010.43.

[43] X. Iturbe, K. Benkrid, C. Hong, A. Ebrahim, R. Torrego, I. Martinez, T. Arslan, and J. Perez, « R3TOS: A Novel Reliable Reconfigurable Real-Time Operating System for Highly Adaptive, Efficient, and Dependable Computing on FPGAs », en, *IEEE Transactions on Computers*, vol. 62, *8*, pp. 1542–1556, Aug. 2013, ISSN: 0018-9340. DOI: 10.1109/TC.2013.79.

[44] N. Harb, S. Niar, M. A. R. Saghir, Y. E. Hillali, and R. B. Atitallah, « Dynamically Reconfigurable Architecture for a Driver Assistant System », en, *in 2011 IEEE 9th Symposium on Application Specific Processors (SASP)*, San Diego, CA, USA: IEEE, Jun. 2011, pp. 62–65, ISBN: 978-1-4577-1212-8. DOI: 10.1109/SASP.2011.5941079.

[45] H. Medeiros, J. Park, and A. Kak, « Distributed object tracking using a cluster-based kalman filter in wireless camera networks », *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, *4*, pp. 448–463, 2008.

[46] X. Li, K. Wang, W. Wang, and Y. Li, « A multiple object tracking method using Kalman filter », *in The 2010 IEEE international conference on information and automation*, IEEE, 2010, pp. 1862–1866.

[47] S.-T. Park and J. G. Lee, « Improved Kalman filter design for three-dimensional radar tracking », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, *2*, pp. 727–739, 2001.

[48] K. Ramachandra, *Kalman Filtering Techniques for Radar Tracking*. CRC Press, 2018.

[49] S. S. Blackman, « Multiple hypothesis tracking for multiple target tracking », *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, *1*, pp. 5–18, 2004.

[50] A. F. Genovese, « The interacting multiple model algorithm for accurate state estimation of maneuvering targets », *Johns Hopkins APL technical digest*, vol. 22, *4*, pp. 614–623, 2001.

[51] J. Ville, *Theory and Applications of the Notion of the Analytic Signal*, English. 1948.

[52] « Chapter 17: Doppler Processing », en, *in Principles of Modern Radar. Vol.1: Basic Principles*, M. A. Richards, J. A. Scheer, and W. A. Holm, Eds., first published 2010, reprinted with corrections 2015, Raleigh, NC: SciTech Publ, 2015, ISBN: 978-1-891121-52-4.

[53] D. Lush, « Airborne Radar Analysis Using the Ambiguity Function », *in IEEE International Conference on Radar*, May 1990, pp. 600–605. DOI: 10.1109/RADAR.1990.201112.

[54] A. Klilou, S. Belkouch, P. Elleaume, P. Le Gall, F. Bourzeix, and M. M. Hassani, « Real-Time Parallel Implementation of Pulse-Doppler Radar Signal Processing Chain on a Massively Parallel Machine Based on Multi-Core DSP and Serial RapidIO Interconnect », en, *EURASIP Journal on Advances in Signal Processing*, vol. 2014, *1*, p. 161, Dec. 2014, ISSN: 1687-6180. DOI: 10.1186/1687-6180-2014-161.

[55] M. Inggs, A. van der Byl, and C. Tong, « Commensal Radar: Range-Doppler Processing Using a Recursive DFT », en, *in 2013 International Conference on Radar*, Adelaide, Australia: IEEE, Sep. 2013, pp. 292–297, ISBN: 978-1-4673-5178-2 978-1-4673-5177-5. DOI: 10.1109/RADAR.2013.6652001.

[56] H.-I. Shin, B.-S. Lee, B.-G. Choi, S.-W. Lee, and W.-W. Kim, « Timing Analysis of Doppler Filter Bank with Parallel Processing Configuration », en, p. 4, 2003.

[57] G. San Antonio, D. R. Fuhrmann, and F. C. Robey, « MIMO Radar Ambiguity Functions », *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, *1*, pp. 167–177, Jun. 2007, ISSN: 1941-0484. DOI: 10.1109/JSTSP.2007.897058.

[58] C.-S. Choi and H. Lee, « An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform », en, *in 2006 First International Conference on Communications and Electronics*, Hanoi, Vietnam: IEEE, Oct. 2006, pp. 352–355, ISBN: 978-1-4244-0568-8 978-1-4244-0569-5. DOI: 10.1109/CCE.2006.350791.

[59] A. Otero, E. De La Torre, T. Riesgo, T. Cervero, S. Lopez, G. Callico, and R. Sarmiento, « Run-Time Scalable Architecture for Deblocking Filtering in H.264/AVC-SVC Video Codecs », en, *in 2011 21st International Conference on Field Programmable Logic and Applications*, Chania: IEEE, Sep. 2011, pp. 369–375, ISBN: 978-1-4577-1484-9. DOI: 10.1109/FPL.2011.72.

[60] M. Feilen, M. Ihmig, A. Zahlheimer, and W. Stechele, « Real-Time Signal Processing on Low-Cost-FPGAs Using Dynamic Partial Reconfiguration », en, *in 2011 International Symposium on Integrated Circuits*, Singapore, Singapore: IEEE, Dec. 2011, pp. 110–113, ISBN: 978-1-61284-865-5 978-1-61284-863-1 978-1-61284-864-8. DOI: 10.1109/ISICir.2011.6131921.

[61] R. Klemm, *Principles of space-time adaptive processing*, 12. IET, 2002.

[62] J. Guerci, J. Goldstein, and I. Reed, « Optimal and Adaptive reduced Rank STAP », *IEEE Transaction on aerospace and electronic systems*, vol. 36, pp. 647–663, 2000.

[63] H Wang and L Cai, « On adaptive Spatial-Temporal processing for airborne Surveillance Radar Systems », *IEEE Transaction on aerospace and electronic systems*, vol. 30, pp. 660–670, 1994.

[64] A. Haimovich and Y Bar-Ness, « An eigenanalysis Interference Canceller », *IEEE Transactions on signal processing*, vol. 39, pp. 76–84, 1991.

[65] A. Haimovich, « The eigencanceler: Adaptive Radar by eigenanalysis methods », *IEEE Transaction on aerospace and electronic systems*, vol. 32, pp. 532–542, 1996.

[66] A. Haimovich and M Berin, « Eigenanalysis-based Space Time Adaptive Radar: Performance analysis », *IEEE Transaction on aerospace and electronic systems*, vol. 33, pp. 1170–1179, 1997.

[67] N. E. Karaoui, « Spectrum estimation for large dimensional covariance matrices using random matrix theory », *The Annals of Statistics*, vol. 36, 2757–2790, 2008.

[68] D. A. Holdsworth and G. A. Fabrizio, « HF interference mitigation using STAP with dynamic degrees of freedom allocation », *in 2008 International Conference on Radar*, IEEE, 2008, pp. 317–322.

[69] W. Zhang, Z. He, J. Li, H. Liu, and Y. Sun, « A Method for Finding Best Channels in Beam-Space Post-Doppler Reduced-Dimension STAP », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, *1*, pp. 254–264, 2014. DOI: 10.1109/TAES.2013.120145.

[70] L. Xie, Z. He, J. Tong, and W. Zhang, « A Recursive Angle-Doppler Channel Selection Method for Reduced-Dimension Space-Time Adaptive Processing », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, *5*, pp. 3985–4000, 2020. DOI: 10.1109/TAES.2020.2983533.

[71] C. Beckhoff, D. Koch, and J. Torresen, « Go Ahead: A Partial Reconfiguration Framework », en, *in 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, Toronto, ON, Canada: IEEE, Apr. 2012, pp. 37–44, ISBN: 978-1-4673-1605-7 978-0-7695-4699-5. DOI: 10.1109/FCCM.2012.17.

[72] R. Zamacola, A. Garcia Martinez, J. Mora, A. Otero, and E. de La Torre, « IMPRESS: Automated Tool for the Implementation of Highly Flexible Partial Reconfigurable Systems with Xilinx Vivado », en, *in 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico: IEEE, Dec. 2018, pp. 1–8, ISBN: 978-1-72811-968-7. DOI: 10.1109/RECONFIG.2018.8641703.

[73] A. Schallenberg, W. Nebel, A. Herrholz, P. A. Hartmann, and F. Oppenheimer, « OSSS+R: A Framework for Application Level Modelling and Synthesis of Reconfigurable Systems », *in Automation Test in Europe Conference Exhibition 2009 Design*, Apr. 2009, pp. 970–975. DOI: 10.1109/DATE.2009.5090805.

[74] Xilinx, *Vitis Unified Software Platform Documentation: Embedded Software Development*, en, 2020.

[75] N. Brown, « Weighing Up the New Kid on the Block: Impressions of Using Vitis for HPC Software Development », en, p. 6, 2020.

[76] J. Ferber, « Les Systèmes Multi-Agents: Vers Une Intelligence Collective », 1995.

[77] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[78] Y. Chen, Z.-L. Liu, and Y.-B. Xie, « A Multi-Agent-Based Approach for Conceptual Design Synthesis of Multi-Disciplinary Systems », en, *International Journal of Production Research*, vol. 52, *6*, pp. 1681–1694, Mar. 2014, ISSN: 0020-7543, 1366-588X. DOI: 10.1080/00207543.2013.848041.

[79] J. Garland and D. Gregg, « Low complexity multiply-accumulate units for convolutional neural networks with weight-sharing », *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, *3*, pp. 1–24, 2018.

[80] D. Strigl, K. Kofler, and S. Podlipnig, « Performance and scalability of GPU-based convolutional neural networks », *in 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, IEEE, 2010, pp. 317–324.

[81] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, « Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA? », *in 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, IEEE, 2012, pp. 232–239.

[82] L. Cork, « Aircraft Dynamic Navigation for Unmanned Aerial Vehicles », en, PhD thesis, Queensland University of Technology, May 2014.

[83] « Chapter 16: Constant False Alarm Rate Detectors », en, *in Principles of Modern Radar. Vol.1: Basic Principles*, M. A. Richards, J. A. Scheer, and W. A. Holm, Eds., first published 2010, reprinted with corrections 2015, Raleigh, NC: SciTech Publ, 2015, ISBN: 978-1-891121-52-4.

[84] Open Source Robotics Foundation, *Gazebo*, en, http://gazebosim.org/.

[85] P. Swerling, « Probability of Detection for Fluctuating Targets », en, *IEEE Transactions on Information Theory*, vol. 6, *2*, pp. 269–308, Apr. 1960, ISSN: 0018-9448. DOI: 10.1109/TIT.1960.1057561.

[86] Julien Mazuet, *HIL Demonstration of the DFT/FFT Self-Adaptive Radar Architecture*, en, https://osf.io/qsz8e/?view_only=8660442e37bc4904815a2deed6385e3e, Mar. 2020.

[87] D. Aalfs, « Principles of modern radar: advanced techniques », *in*. SciTech Pub., 2013, ch. 9.3 Adaptive Jammer Cancellation.

[88] J Xu, S Zhu, and G Liao, « Space-Time-Range adaptive processing for airborne radar systems », *IEEE sensors journal*, vol. 15, *4*, pp. 1602–1610, 2015.

[89] J. Lundén and V. Koivunen, « Deep learning for HRRP-based target recognition in multistatic radar systems », *in 2016 IEEE Radar Conference (RadarConf)*, IEEE, 2016, pp. 1–6.

[90] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, « Reinforcement learning for adaptable bandwidth tracking radars », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, *5*, pp. 3904–3921, 2020.

[91] S. Bidon, O. Besson, and J.-Y. Tourneret, « Knowledge-aided STAP in heterogeneous clutter using a hierarchical Bayesian algorithm », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, *3*, pp. 1863–1879, 2011.

[92] M. Riedl and L. C. Potter, « Knowledge-aided Bayesian space-time adaptive processing », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, *4*, pp. 1850–1861, 2018.

[93] F. Tao, T. Wang, J. Wu, and X. Lin, « A novel KA-STAP method based on Mahalanobis distance metric learning », *Digital Signal Processing*, vol. 97, p. 102 613, 2020.

[94] I. S. Reed, J. D. Mallett, and L. E. Brennan, « Rapid convergence rate in adaptive arrays », *IEEE Transactions on Aerospace and Electronic Systems*, *6*, pp. 853–863, 1974.

[95] K. Duan, H. Xu, H. Yuan, H. Xie, and Y. Wang, « Reduced-DOF Three-Dimensional STAP via Subarray Synthesis for Nonsidelooking Planar Array Airborne Radar », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, *4*, pp. 3311–3325, 2019.

[96] The ITU Radiocommunication Assembly, « RECOMMENDATION ITU-R P.372-12 », International Telecommunication Union, Tech. Rep., 2015.

[97] R Kedzierawski, J. Le Caillec, and W Czarnecki, « Simulation of subsurface imaging for remote sensing and buried object detection from airborne platform », *European Journal of Remote Sensing*, vol. 52, pp. 583–598, 2019.

[98] S. T. Smith, « Space-Time clutter covariance matrix computation and interference subspace tracking », *in Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, IEEE, vol. 2, 1995, pp. 1193–1197.

[99] J. Le Caillec, S. Redadaa, C. Sintes, B. Solaiman, and M. Benslama, « Focusing Problems of a Buried Point Scatterer using a Low Frequency SAR », *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, *1*, 2011. DOI: 10.1109/TAES.2011.5705685.

[100] J.-M. Le Caillec, T. Górski, G. Sicot, and A. Kawalec, « Theoretical performance of space-time adaptive processing for ship detection by High-Frequency surface wave radars », *IEEE Journal of Oceanic Engineering*, vol. 43, *1*, pp. 238–257, 2017.

[101] R. May, G. Dandy, and H. Maier, « Review of input variable selection methods for artificial neural networks », *Artificial neural networks-methodological advances and biomedical applications*, vol. 10, p. 16 004, 2011.

[102] T. Dyba and T. Dingsoyr, « What Do We Know about Agile Software Development? », en, *IEEE Software*, vol. 26, *5*, pp. 6–9, Sep. 2009, ISSN: 0740-7459. DOI: 10.1109/MS.2009.145.

[103] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, P.-F. Chiu, R. Avizienis, B. Richards, J. Bachrach, D. Patterson, E. Alon, B. Nikolic, and K. Asanovic, « An Agile Approach to Building RISC-V Microprocessors », en, *IEEE Micro*, vol. 36, *2*, pp. 8–20, Mar. 2016, ISSN: 0272-1732. DOI: 10.1109/MM.2016.11.

[104] J. Dean, « 1.1 The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design », en, *in 2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, San Francisco, CA, USA: IEEE, Feb. 2020, pp. 8–14, ISBN: 978-1-72813-205-1. DOI: 10.1109/ISSCC19947.2020.9063049.

[105] A. Biondi, A. Balsini, M. Pagani, E. Rossi, M. Marinoni, and G. Buttazzo, « A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs », Nov. 2016, pp. 1–12. DOI: 10.1109/RTSS.2016.010.

[106] K. Vipin and S. A. Fahmy, « Automated partitioning for partial reconfiguration design of adaptive systems », *in 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, IEEE, 2013, pp. 172–181.

[107] H. Tang, « Algorithm/Architecture Co-Design for Wireless Communications Systems », en, PhD thesis.

[108] K. Desnos, M. Pelcat, J.-F. Nezan, S. S. Bhattacharyya, and S. Aridhi, « PiMM: Parameterized and Interfaced Dataflow Meta-Model for MPSoCs Runtime Reconfiguration », *in 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, Jul. 2013, pp. 41–48. DOI: 10.1109/SAMOS.2013.6621104.

[109] A. Kamppi, L. Matilainen, J.-M. Määttä, E. Salminen, and T. D. Hämäläinen, « Extending IP-XACT to embedded system HW/SW integration », *in 2013 International Symposium on System on Chip (SoC)*, IEEE, 2013, pp. 1–8.

[110] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, « Run-Time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration », en, *in 2009 International Conference on Field Programmable Logic and Applications*, Prague, Czech Republic: IEEE, Aug. 2009, pp. 498–502. DOI: 10.1109/FPL.2009.5272463.

[111] J. Piat and J. Crenne, « Modeling Dynamic Partial Reconfiguration in the Dataflow Paradigm », *in 2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2014, pp. 1–6. DOI: 10.1109/SiPS.2014.6986103.

[112] M. Auguin and F. Larbey, « OPSILA: an advanced SIMD for numerical analysis and signal processing », *in Microcomputers: developments in industry, business, and education, Ninth EUROMICRO Symposium on Microprocessing and Microprogramming, Madrid, September 13*, vol. 16, 1983, pp. 311–318.

[113] G. Delaval, H. Marchand, and E. Rutten, « Contracts for modular discrete controller synthesis », *in Proceedings of the ACM SIGPLAN/SIGBED 2010 conference on Languages, compilers, and tools for embedded systems*, 2010, pp. 57–66.

[114] C. Boudjennah, B. Combemale, D. Exertier, S. Lacrampe, and M.-A. Peraldi-Frati, « CLARITY: Open-Sourcing the Model-Based Systems Engineering Solution Capella », *in Second Workshop on Open Source Software for Model Driven Engineering (OSS4MDE'15)*, CEUR, 2015.

[115] S. Misra, P. N. Mohammed, B. Guner, C. S. Ruf, J. R. Piepmeier, and J. T. Johnson, « Microwave radiometer radio-frequency interference detection algorithms: A comparative study », *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, *11*, pp. 3742–3754, 2009.

[116] J. Lemmon, « Wideband model of HF atmospheric radio noise », *Radio science*, vol. 36, pp. 1385–1391, 2001.

[117] J. Herman and X. DeAngelis, « Bandwidth expansion effects on the voltage deviation parameter (V_d) of MF and HF atmospheric radio noise », *Radio science*, vol. 22, 1987.

List of Figures

List of Tables

Appendix A

Interference models

In this appendix, we detail, the correlation between pulse and antenna voltage at a given range gate. The angular frequency ω is constant and equal to ω_c for monochromatic wave and it is given by given by $\omega_c + K_\omega t$, for $-\tau/2 \leq t \leq \tau/2$, and the corresponding wavevector norm $k_p = \omega/c$ (constant for monochromatic wave, time dependent for chirps)

Mimicking [100], considering the emission of the first pulse as the time origin, the received voltage at antenna n and for m for a given, elevation θ, that is an isorange or equivalently for the travelling time t_0, is given by

$$v(t) = \int_{-\pi}^{+\pi} B(\phi) e^{j\omega(t+t_0+mT_r)\left(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right)} e^{-jnk_p d\cos(\theta)\sin(\phi)} d\phi \tag{A.1}$$

where $B(\phi)$ is the backscattered field from azimuth ϕ per unit of angle and per unit of time. The range compression is given by the convolution of the received voltage by the reverse chirp $c(-t) = e^{-j\omega t}$ over a time duration τ centered in t_0. Thus, neglecting the

second order term we have

$$
\begin{aligned}
V(m,n) &= \int_{-\tau/2}^{\tau/2} v(t-t_0)e^{-j\omega t}dt \\
&\simeq \int_{-\pi}^{+\pi}\int_{-\tau/2}^{\tau/2} B(\phi)e^{j\omega(t+mT_r)\left(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right)} \\
&\qquad e^{-jnk_pd\cos(\theta)\sin(\phi)}e^{-j\omega t}dtd\phi \\
&= \int_{-\pi}^{+\pi} B(\phi)e^{j\omega_c\left(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right)m.T_r}e^{-jn\frac{\omega_c}{c}d\cos(\theta)\sin(\phi)}. \\
&\int_{-\tau/2}^{\tau/2} e^{j\left(K_\omega mT_r(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c})\right)t}e^{j\left(2.\omega_c\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right)t} \\
&\qquad e^{-jn\frac{K_\omega}{c}d\cos(\theta)\sin(\phi)t}dtd\phi \\
&= \int_{-\pi}^{+\pi} B(\phi)e^{j\omega_c(\left(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right)m.T_r}.e^{-jn\frac{\omega_c}{c}d\cos(\theta)\sin(\phi)} \\
&\qquad 2\frac{\sin(\Omega_C(m,n)\tau/2)}{\Omega_C(m,n)}d\phi
\end{aligned}
\tag{A.2}
$$

with

$$
\begin{aligned}
\Omega_C(m,n) =& mK_\omega T_r\left(1+2\frac{v_a.\cos(\theta)\sin(\phi)}{c}\right) \\
&+ 2.\omega_c\frac{v_a.\cos(\theta)\sin(\phi)}{c} - n\frac{K_\omega}{c}d\cos(\theta)\sin(\phi)
\end{aligned}
\tag{A.3}
$$

by neglecting the second order term. According to the values exposed in section 4.4, we can make the approximations $M_fK_\omega T_r \ll \omega_c$ and $N_f\frac{K_\omega}{c}d \ll \omega_c$, (A.3) becomes independent of m and n and we obtain:

$$
\Omega_C = 2.\omega_c\frac{v_a.\cos(\theta)\sin(\phi)}{c} \tag{A.4}
$$

The results is that obtain for a monochromatic wave (i.e. $K_\omega = 0$).
In order to calculate the clutter covariance matrix entry $R_{(n,m)(n_1,m_1)} = E\{V(m,n).V^H(m_1,n_1)\}$, we have to calculate the expectation of a product of integrals over ϕ and ϕ' (see [98]). By transforming the product into double integral, permuting the integral and the expectation and observation the random backscattered field $B(\phi)$ verifies $E\{B(\phi).B(\phi')\} = |A(\phi)|^2\delta(\phi-\phi')$ and preceding similarly to [98] for a fast computation, we obtain Eq. (4.13)

by observing that $\frac{\omega_c}{c}.d.\cos(\theta) = \pi\cos(\theta) = \pi\sqrt{1-(\frac{h_a}{r\cdot\delta r})^2}$, for $r \geq \frac{h_a}{\delta r}$ with the hypothesis $d = \lambda/2$.

For the RFI interference, we reuses the model of [100], derived from [116], although initially defined in the HF/UHF bands in Eq. (4.14), we have

$$\begin{aligned} J(u,m,m_1) = & \frac{T_Q}{T_B+T_Q}(1-F_q(|u+(m-m_1)T_r|)) \\ & \int_0^{V_T} v^2 f_V(v)dv \\ & + \frac{T_B}{T_B+T_Q}(1-F_b(|u+(m-m_1)T-r|)) \\ & \int_{V_T}^{+\infty} v^2 f_V(v)dv \end{aligned} \tag{A.5}$$

where F_q and F_p are the CDF of the quite and burst period given in the two cases by

$$F_{|b,q}(t) = 1 - e^{-\frac{C_1}{C_2}\left(1-e^{-C_2 t}\right)-C_3 t} \tag{A.6}$$

and $T_B = 26ms$ and $T_Q = 247ms$ leading the constants $C_1 = 57.43$, $C_2 = 32.23$ and $C_3 = 12.68$ for burst and $C_1 = 18.62$, $C_2 = 16.62$ and $C_3 = 1.49$ quite period. The voltage envelop pdf in Eq. (A.5) being given by

$$f_V(V) = \frac{(\mu-1)\gamma^{\mu-1}V}{(V^2+\gamma^2)^{(\mu+1)/2}} \tag{A.7}$$

in Eq. (A.6). V_T is threshold below which, we consider to be a in quiet period, above which we have a burst i.e $\int_0^{V_T} f_V(V)dV = \frac{T_Q}{T_B+T_Q}$. In Eq. (A.7), the parameter μ is related to V_d (see Fig. 3 of [116]) while γ is related to the power of the voltage envelope (i.e. $P_{RFI} = 2\gamma^2/(\mu-3)$). Herman and DeAngelis [117] proposed the law :

$$V_d = 0.609V_0 + (0.910 + 0.250V_0)\log(B) \tag{A.8}$$

where B is the waveform bandwidth and V_0 is the reference of V_d for a bandwidth of 200 Hz. As observed in Eq. (A.7), the voltage envelope pdf exhibits a heavy tail (due to burst) and the RFI statistics are non Gaussian. For chirp waveform, the compression factor $\text{sinc}(K_\omega.\tau/2)$ has to be added, but as previously stated we can approximate this factor by the unity.

For the broadband jammer interference, we consider the case where the signal is uncorrelated from one pulse to another. The received voltage after range compression is given by

$$\begin{aligned} V(m,n) &= \int_{-\tau/2}^{\tau/2} A_J e^{-jnk_p d\cos(\theta_J)\sin(\phi_J)} C(m) e^{-j\omega t} dt \\ &= A_J e^{-jn\frac{\omega_c}{c} d\cos(\theta_J)\sin(\phi_J)} C(m) \\ &\qquad \int_{-\tau/2}^{\tau/2} e^{-jn\frac{K_\omega}{c} d\cos(\theta_J)\sin(\phi_J)t} e^{-j\omega_c t} e^{-jK_\omega t^2} dt \\ &= A_J.\tau.e^{-jn\frac{\omega_c}{c} d\cos(\theta_J)\sin(\phi_J)} C(m)\,\mathrm{sinc}(\Omega_J(n)) \end{aligned} \tag{A.9}$$

with

$$\Omega_J(n) = \omega_c + n\frac{K_\omega}{c} d\cos(\theta_J)\sin(\phi_J) \tag{A.10}$$

As in the previous cases, we can neglect the second term of $\Omega_J(n)$ and then retrieve the monochromatic case:

$$\Omega_J = \omega_c \tag{A.11}$$

A_J is the random amplitude of the jammer signal with variance σ_J^2 ($P_J = \sigma_J^2$). A_j remains consnat during all the radar observation. Thus, $C(m)$ is a random complex number for each index m, with $|C(m)| = 1$ and $\arg(C(m)) \sim U(0, 2\pi)$. $C(m)$ reflects the uncorrelation of the signal in the slow-time domain, which makes the inclusion of the Doppler irrelevant in the broadband jammer model.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Université de Bretagne Occidentale

Titre : Reconfiguration des ressources matérielles et logicielles d'un système radar embarqué en mission d'interception

Mot clés : Radar, traitement du signal numérique, reconfiguration dynamique partielle

Résumé : Les systèmes radar embarqués sont limités en ressources de calcul. Dans le même temps, ces systèmes doivent utiliser des algorithmes toujours plus complexes et demandant de plus en plus de ressources de calcul. De plus, les radars modernes doivent être capables d'utiliser plusieurs modes de fonctionnement durant une même mission. Dans un système classique, les ressources sont attribués au moment de la conception du système et un compromis doit être trouvé entre la qualité des algorithmes implémentés et leur consommation en ressources, toujours au détriment des performances de traitement. Afin d'obtenir de meilleurs performances, il est possible de réattribuer les ressources de calculs au cours de la mission par reconfiguration dynamique.

Ces travaux de thèse se consacrent dans un premier temps à optimiser des applications de radars embarqués par le biais de la reconfiguration dynamique. Deux architectures matérielles reconfigurables, ainsi qu'un algorithme pouvant améliorer les performances de traitements radar sont présentés. Dans un second temps, partant du constat que les systèmes reconfigurables sont difficiles à développer car ils doivent prendre en compte des aspects algorithmiques, matériels et logiciels, une méthodologie de développement pluridisciplinaire est proposée pour faciliter le développement de systèmes reconfigurables efficaces.

Title: Reconfiguration of the hardware and software resources of an embedded radar system in an interception mission

Keywords: Radar, digital signal processing, dynamic partial reconfiguration

Abstract: Embedded radar systems are limited in computing resources. At the same time, these systems must use increasingly complex algorithms that require more and more computing resources. In addition, modern radars must use several operating modes during a single mission. In a classical system, the resources are allocated at the system design time and a compromise must be found between the implemented algorithms quality and their resources consumption, always at the expense of processing performance. In order to obtain better performances, it is possible to reallocate the computational resources during the mission by dynamic reconfiguration.

This thesis work is first dedicated to the optimization of onboard radar applications through dynamic reconfiguration. Two reconfigurable hardware architectures, as well as an algorithm that can improve the performance of radar processing are presented. In a second step, starting from the fact that reconfigurable systems are difficult to develop because they must take into account algorithmic, hardware and software aspects, a multidisciplinary development methodology is proposed to facilitate the efficient reconfigurable systems development.