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

Julien Mazuet, Michel Narozny, Catherine Dezan, Jean-Philippe Diguet

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

Julien Mazuet[1,2], Michel Narozny[1], Catherine Dezan[2], Jean-Philippe Diguet[2]
[1]*Thales LAS-France*, Élancourt, France, [2]*Lab-STICC, CNRS, UBO / UBS*, Brest / Lorient, France
julien.mazuet@univ-ubs.fr

***Abstract*—Radar is one of the domains where adaptability is paramount and algorithms must be adapted to system state. However, most systems include static implementations on FPGA or ASIC to process the massive amount of data from multiple sensors in parallel. The classic approach is to configure hardware logic through registers to switch radar modes, requiring to hardwire all configurations. In embedded systems, FPGA dynamic partial reconfiguration (DPR) is a promising solution to reuse scarce resources. In this paper, we use DPR for radar processing in order to switch between a classic discrete Fourier transform (DFT) sum and a fast Fourier transform (FFT) to enhance Doppler extraction. Our study 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. Finally, we provide a case study and a hardware-in-loop simulation with a reconfigurable radar implementation.**

***Index Terms*—FPGA, DPR, radar, QoS, DFT**

I. Introduction

Modern embedded radar applications require high performance architectures. This point is specially critical in small aircraft and unmanned aerial vehicles. But a radar system can inherently leverage data parallelism, which can be efficiently implemented on a FPGA [1]. For instance, active electronically scanned array (AESA) radar have multiple channels which can be processed independently by HW accelerators [2]. Besides, these applications have strong performance and latency constraints. Furthermore, a radar system has different modes [3], like high pulse recurrence frequency (PRF) and low PRF, or more complex modes like search and tracking or air-air and air-ground. These modes have different motivations and requirements. The mode adaptation is often ensured by runtime software adaptation and hardware reconfigurations through dedicated registers. However, this approach means that configurations are simultaneously implemented within the same programmable logic (PL). Important configuration changes result in the PL overuse. Consequently, designers must limit the performances of the target algorithms to fit with available PL resources. In that perspective, using DPR appears as a promising solution. However, this approach requires a specific methodology to allow an efficient collaboration between application experts who can determine the algorithm choices and specify the reconfiguration decision making with hardware experts who can design highly optimized architectures using CAD tools. In this work we apply this methodology to design the first solution that fully benefit of FFT and DFT compliant configurations to mitigate the tradeoff between resolution and latency according to user requirements.

Section II presents a state of the art on DFT in the radar domain, and the use of reconfiguration. Section III details the DFT algorithm in the context of radar systems and explains the two main methods to compute it. Section IV proposes a methodology to take efficient reconfiguration decisions at runtime. A methodology to create an adaptive reconfigurable DFT for radar, is described in section V. Section VI shows and comments the results obtained through a hardware in the loop (HIL) implementation on a case study. Section VII concludes the paper and draws directions for future work.

II. State of the art

Reconfigurable FPGA have been strongly used to efficiently implement signal processing applications [4] but few works have addressed the question of radar [5]–[7]. 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 [8]). 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 [9]. The main methods are the FFT [10], [11] and the classic DFT summation [12], [13]. 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 [14]. 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 [15]–[17], permits to overcome this limitation.

The first contribution of this paper is 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.

III. DFT options for radar processing

A. DFT algorithm

The DFT algorithm (defined by Eq. (1)) is used to extract the spectral distribution of a discrete signal.

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

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. (1). However, in some applications, this direct DFT implementation can be useful to order certain operations.

B. 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. 1-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 [10], [11].

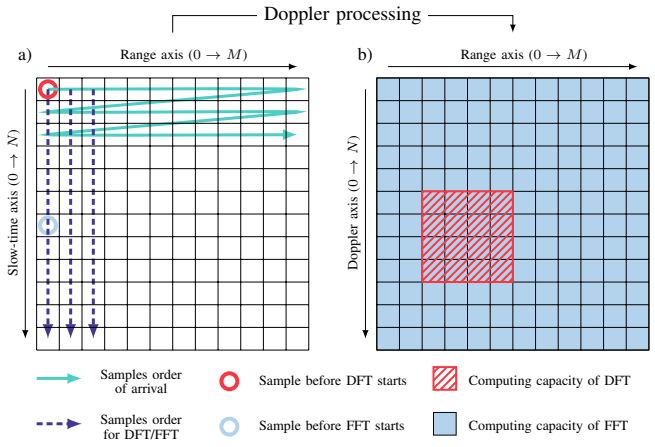

Fig. 1. Constitution of a range-Doppler radar map. a) down-sampling and DFT. b) Representation of the computing capacities of DFT and FFT.

Fig. 1 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.

C. 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 paper, 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. 1-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.(1) 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

D. 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 free 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

E. 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. 1-b).

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

IV. 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) [18]. 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. (2).

$$V(k) = \begin{cases} 1, \text{if } \ell(k) \geqslant \gamma \\ 0, \text{if } \ell(k) < \gamma \end{cases} \quad (2) \qquad L_q(k+q) = \frac{1}{q}\sum_{i=1}^{q} V(k+i) \quad (3)$$

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). 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. (4, 5) describe the state of the configuration with regard to the QoS criterion. Fig. 2 gives another representation of this system through a cyclic graph and a transition hysteresis.

$$s_k = \begin{cases} S_{DFT}, & \text{if } S_{DFT} \cdot \bar{a} + S_{FFT} \cdot b \\ S_{FFT}, & \text{if } S_{DFT} \cdot a + S_{FFT} \cdot \bar{b} \end{cases} \quad (4)$$

Where: $$a = \bar{b} = f_{QoS}(k+q) < \lambda \quad (5)$$

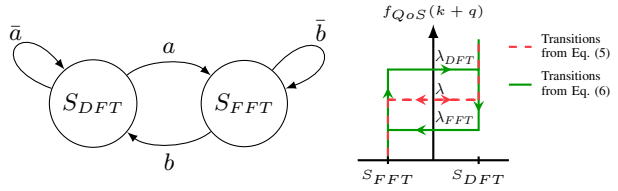

Fig. 2. Control graph and associated transition hysteresis

A transition based on Eq. (5) 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. (6).

$$a = f_{QoS}(k+q) \leqslant \lambda_{FFT} \; ; \; b = f_{QoS}(k+q) \geqslant \lambda_{DFT} \quad (6)$$

Fig. 2 shows that Eq. (6) 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.

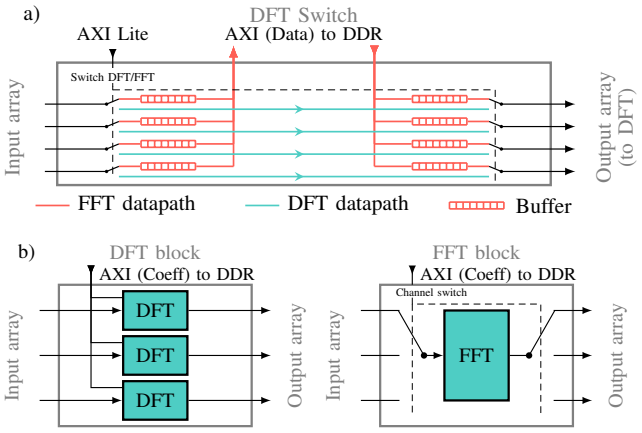

Fig. 3. a) Switch block to adapt input data ordering for DFT. b) The two configurations of the DFT IP.

V. Implementation of a reconfigurable DFT

A. 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. 4 shows a sixteen sensors radar system which implements DFT reconfiguration, where each RP performs four DFT to divide the reconfiguration time in four steps. During this process, the radar system is still running with $3/4$ of the channels.

B. Data reordering

As introduced in Sec. III, 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-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.

C. Architectures of DFT/FFT configurations

An efficient DPR requires that the configurations sharing the same RP have similar amount of resources since the RP

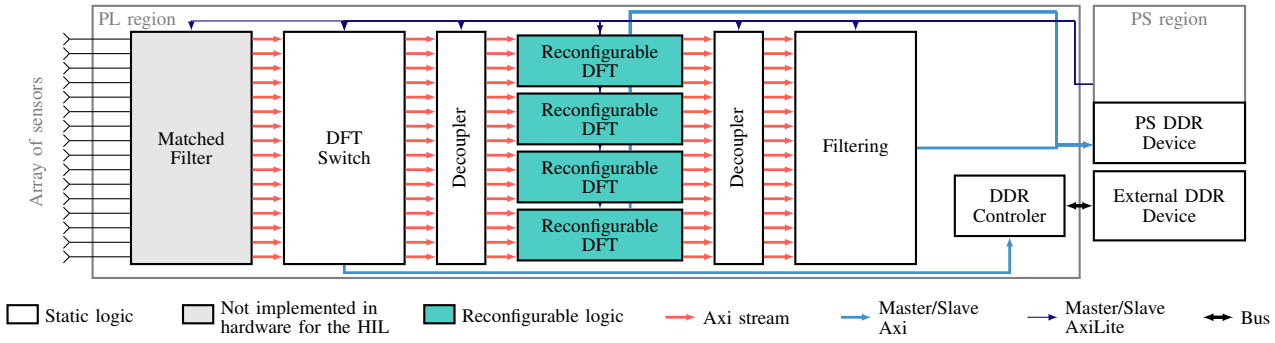

Fig. 4. Architecture used for the final results

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-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.

D. 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 I as the *HLS estimation quality*. Our HLS code allows to choose the number of channels per FFT in the reconfigurable IP.

VI. Case Study

A. System description

To study the performance of our architecture, we implement the system described in Fig. 4 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. 4. 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 [19] (Cell-Averaging Constant False Alarm Rate) implemented in software. The reconfiguration controller defined in Section IV is implemented in software as well.

We use the Gazebo simulator [20] 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. IV) which is updated accordingly. The board sends the output signal and the detection plots to our Python 3 interface through Ethernet, for graphical plots.

B. 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 flush operation out of the reconfiguration time. The impact on performance is given in the Sec. VI-E.

C. 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 paper, 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

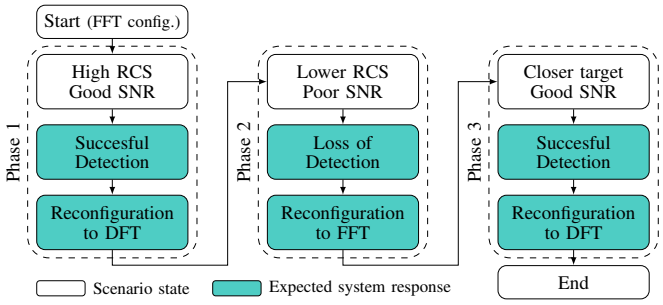

Fig. 5. Case study scenario

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 [21].

The scenario is composed of three phases depicted in Fig. 5. 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 [22].

D. Implementation results

As we stated in Sec. V-D, 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 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 I). 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.

TABLE I
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

The full architecture is depicted in Fig. 4. 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. V. 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.

E. Performance results

With this configuration, the observed system response meets the expected response described in Sec. VI-C. 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.

VII. CONCLUSION

To the best of our knowledge this paper 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.IV, 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.

REFERENCES

[1] Ming Yang, Jing Yang, Yanan Hou, and Cheng Jin. Implementation architecture of signal processing in pulse Doppler radar system based on FPGA. *The Journal of Engineering*, 2019(21):7335–7338, November 2019.

[2] D. Govind Rao, Aalhad P. Deshpande, N. S. Murthy, and A. Vengadarajan. Digital beam former architecture for sixteen elements planar phased array radar. In *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pages 532–537, Konya, Turkey, May 2013. IEEE.

[3] Eric Chamouard. Radars aéroportés multifonctions (in french with english abstract). page 24, 2013.

[4] Russell Tessier, Kenneth Pocek, and Andre DeHon. Reconfigurable Computing Architectures. *Proceedings of the IEEE*, 103(3):332–354, March 2015.

[5] Emmanuel Seguin, Russell Tessier, Eric Knapp, and Robert W. Jackson. A Dynamically-Reconfigurable Phased Array Radar Processing System. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 258–263, Chania, Greece, September 2011. IEEE.

[6] Yuxi Zhang, Zhanchao Wang, and Jun Wang. Integrated radar signal processing using FPGA dynamic reconfiguration. In *2016 CIE International Conference on Radar (RADAR)*, pages 1–4, Guangzhou, China, October 2016. IEEE.

[7] Naim Harb, Smail Niar, Mazen A. R. Saghir, Yassin El Hillali, and Rabie Ben Atitallah. Dynamically reconfigurable architecture for a driver assistant system. In *2011 IEEE 9th Symposium on Application Specific Processors (SASP)*, pages 62–65, San Diego, CA, USA, June 2011. IEEE.

[8] J. Ville. *Theory and Applications of the Notion of the Analytic Signal*. 1948.

[9] Chapter 17: Doppler Processing. In Mark A. Richards, James A. Scheer, and William A. Holm, editors, *Principles of Modern Radar. Vol.1: Basic Principles*. SciTech Publ, Raleigh, NC, first published 2010, reprinted with corrections 2015 edition, 2015.

[10] D.C. Lush. Airborne radar analysis using the ambiguity function. In *IEEE International Conference on Radar*, pages 600–605, May 1990.

[11] Abdessamad Klilou, Said Belkouch, Philippe Elleaume, Philippe Le Gall, François Bourzeix, and Moha M'Rabet 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. *EURASIP Journal on Advances in Signal Processing*, 2014(1):161, December 2014.

[12] Michael Inggs, Andrew van der Byl, and Craig Tong. Commensal radar: Range-Doppler processing using a recursive DFT. In *2013 International Conference on Radar*, pages 292–297, Adelaide, Australia, September 2013. IEEE.

[13] Hyun-Ik Shin, Bum-Suk Lee, Beyung-Gwan Choi, Seok-Woo Lee, and Whan-Woo Kim. Timing Analysis of Doppler Filter Bank with Parallel Processing Configuration. page 4, 2003.

[14] Geoffrey San Antonio, Daniel R. Fuhrmann, and Frank C. Robey. MIMO Radar Ambiguity Functions. *IEEE Journal of Selected Topics in Signal Processing*, 1(1):167–177, June 2007.

[15] Chang-Seok Choi and Hanho Lee. An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform. In *2006 First International Conference on Communications and Electronics*, pages 352–355, Hanoi, Vietnam, October 2006. IEEE.

[16] 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. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 369–375, Chania, September 2011. IEEE.

[17] Michael Feilen, Matthias Ihmig, Anton Zahlheimer, and Walter Stechele. Real-time signal processing on low-cost-FPGAs using dynamic partial reconfiguration. In *2011 International Symposium on Integrated Circuits*, pages 110–113, Singapore, Singapore, December 2011. IEEE.

[18] Yaakov Bar-Shalom and Fred Daum. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82–100, December 2009.

[19] Chapter 16: Constant False Alarm Rate Detectors. In Mark A. Richards, James A. Scheer, and William A. Holm, editors, *Principles of Modern Radar. Vol.1: Basic Principles*. SciTech Publ, Raleigh, NC, first published 2010, reprinted with corrections 2015 edition, 2015.

[20] Open Source Robotics Foundation. Gazebo. http://gazebosim.org/.

[21] P. Swerling. Probability of detection for fluctuating targets. *IEEE Transactions on Information Theory*, 6(2):269–308, April 1960.

[22] Julien Mazuet. HIL demonstration of the DFT/FFT self-adaptive radar architecture. https : / / osf . io / qsz8e / ?view_only = 8660442e37bc4904815a2deed6385e3e, March 2020.