International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 6, December 2020, pp. 6541~6548
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6541-6548

Parallel implementation of pulse compression method on a multi-core digital signal processor

Abdessamad Klilou[1], Assia Arsalane[2]
[1]Department of Electrical Engineering, Laboratory of Automation, Conversion of Energy and Microelectronic (LACEM), Sultan Moulay Slimane University, Morocco
[2]Laboratory of Engineering and Applied Technologies, High School of Technologies, Sultan Moulay Slimane University, Morocco

Article Info

Article history:

Received Feb 26, 2020
Revised May 30, 2020
Accepted Jun 15, 2020

Keywords:

Multi-core DSP
OpenMP
Pulse compression
Radar
Real time processing

ABSTRACT

Pulse compression algorithm is widely used in radar applications. It requires a huge processing power in order to be executed in real time. Therefore, its processing must be distributed along multiple processing units. The present paper proposes a real time platform based on the multi-core digital signal processor (DSP) C6678 from Texas Instruments (TI). The objective of this paper is the optimization of the parallel implementation of pulse compression algorithm over the eight cores of the C6678 DSP. Two parallelization approaches were implemented. The first approach is based on the open multi processing (OpenMP) programming interface, which is a software interface that helps to execute different sections of a program on a multi core processor. The second approach is an optimized method that we have proposed in order to distribute the processing and to synchronize the eight cores of the C6678 DSP. The proposed method gives the best performance. Indeed, a parallel efficiency of 94% was obtained when the eight cores were activated.

Corresponding Author:

Abdessamad Klilou,
Department of Electrical Engineering,
Laboratory of Automation, Conversion of Energy and Microelectronic (LACEM),
Faculty of Sciences and Technologies, Sultan Moulay Slimane University,
523, Beni Mellal, Morocoo.
Email: a.klilou@usms.ma

1. INTRODUCTION

Pulse compression algorithm is widely used in radar applications, such as pulse Doppler radar [1], ground-moving target indicator (GMTI) [2], and synthetic aperture radar (SAR) [3]. It is carried out on the acquired signal in order to extract distance of target from radar with high precision. Its major constraints is that it requires a high-computing power. Consequently, one processing element cannot holds its processing in real-time. Therefore, one solution is using multiple computing cores working together; each one of them execute a small portion of processing.

This paper presents the C6678 DSP from TI as a processing platform. It provides a high performance floating-point calculation with a low power consumption. In fact, it contains eight independent C66x cores, each core run to a frequency of 1GHz. Moreover, it provides a maximum performance of 128 GFLOPS for a single precision floating point calculation [4]. In addition, several research communities have developed high-performance computing systems using the C6678 DSP [3, 5-9].

Embedded systems based on DSP has proved its efficiency to execute a large number of signal processing algorithm in real time. It has been used by a large scientific community to build real time embedded systems. Abdelkareem et al. [10] have developed high performance software that requires real-time embedded systems for emerging technology areas like 5G Wireless and software defined

networking (SDN). Arsalane et al., [11-15] have developed an embedded system based on the C6678 DSP for beef meet freshness evaluation.

In our previous works [1], we presented a real time parallel implementation of pulse Doppler radar signal processing chain, including beam forming, pulse compression and Doppler, on a parallel machine with 2 C6678 DSPs boards (a total of 16 processing cores). A straightforward model has been used and optimized as a processing parallelization strategy. All communications, including data exchange and synchronization, between processing DSP cores goes through the inter-processor communication bus Serial RapidIO (SRIO), which we have optimized its use [16, 17]. The major obtained result is a parallel efficiency of about 90%.

Huang et al., [18] have proposed a parallel implementation of beam forming algorithm on TI-based Tomahawk platform containing six DSP cores. The algorithm is widely used in radar applications. In fact Huang et al., [18] have used the OpenMP interface [19] to distribute the processing over the six DSP cores. Results show a maximum speedup about 3.7. Mego et al., [20] have evaluated the performance of parallelization of basics signal processing algorithms, such as finite impulse response (FIR) filter, discrete fourier transform (DFT) and fast fourier transform (FFT), on the C6678 DSP. In their study, authors have used the OpenMP interface to distribute the processing over the eight DSP cores. Obtained results show that the relative speedup is highly dependent on the algorithm and the amount of processed data. Results show a maximum speedup of about 6. Yu et al., [21] have implemented the pulse Doppler radar signal processing chain on computing platform based on the C6678 DSP. The studied algorithm include three steps: beam forming, pulse compression and Doppler filtering. They have used OpenMP framework for parallel implementation. Obtained results show that multi-threaded execution is less than single-threaded. According to authors, this difference was explained by the highly non-linear memory accesses required by the FFT and the inverse fast fourier transform (IFFT). Wang et al. [3] have implemented and optimized SAR algorithms on the eight core of the C6678 DSP. The studied algorithm include two steps of pulse compression method (range compression and azimuth compression), range cell migration correction (RCMC) and corner turn. The OpenMP framework was used to instantiate individual threads across the eight cores. Obtained results show that the timing required for range compression and azimuth compression scales very well with the increase of the number of operational cores. However, the other RCMC and corer turn steps saturates at around four cores. For the total execution time, the acceleration factor with eight cores relative to a single core is equal to 5.6.

From all presented researches works, OpenMP has been successfully tested to distribute many signal-processing algorithms over multi-core DSP platforms. However, the obtained parallel efficiency does not exceed 70% in the best cases. In this paper, an optimized method is proposed as an alternative to OpenMP method in order to improve the performances.

The major contribution of this paper is the distribution of the pulse compression algorithm over the eight processing core of the C6678 DSP. We have implemented two parallelization approaches. The first one, is based one the OpenMP, which is a shared-memory application programming interface (API) whose features, are based on prior efforts to facilitate shared-memory parallel programming. As the C6678 DSP integrates two levels of memory shared between the eight cores, which are the internal multi-core shared memory (MSM) and the external DDR memory, the OpenMP is fully adapted. The second approach is an optimized method that we have proposed to distribute the processing of the pulse compression algorithm on the eight cores. The performance of the two parallelization methods are compared to each other based on speedup and parallel efficiency indicators.

This paper is organized as follows. Section 2 presents an overview of pulse compression method, experimental platform, and metrics used for evaluating parallel processing performance. Moreover, it presents the proposed mehod to distribute pulse compression algorithm on multiples cores. Section 3 provides the experimental results of parallel implementation of pulse compression using the OpenMP API and the proposed approach. Finally, a conclusion is provided in section 4.

2. RESEARCH METHOD

2.1. Pulse compression algorithm

A convolution operation between the transmitted and the received pulse is performed in order to detect radar targets [22]. In fact, two closely targets are fully merged in case where the wave sent by the radar is a sinusoidal signal as shown in Figure 1. To improve detection accuracy of closely targets, the transmitted wave undergoes a linear frequency modulation operation shown in Figure 2(b). The obtained signal is called ‘Chirp’ shown in Figure 2(a).

To optimize the processing of the pulse compression, the convolution operation is realized in the frequency space. It is carried out by performing the product of the FFT [23, 24] of the input signal and the pulse compression coeficients followed by the IFFT in order to return to the time domain as shown in

Figure 3. Its computing complexity depends on FFT, inverse IFFT, and point-wise vector multiplication. The complexity of computing radix-2 FFT is equal to $5Nlog_2(N)$ floating-point operations; N is the FFT size and must be a power of two. The complexity of computing the IFFT is the same as for the FFT. For the point-wise vector multiplication, $6N$ floating-point operations are needed. Therefore, the throughput of the pulse compression in the frequency domain is equal to $(10Nlog_2(N) + 6N)/T$ FLOPS, in which N is the number of range gates and T is the execution time in second.

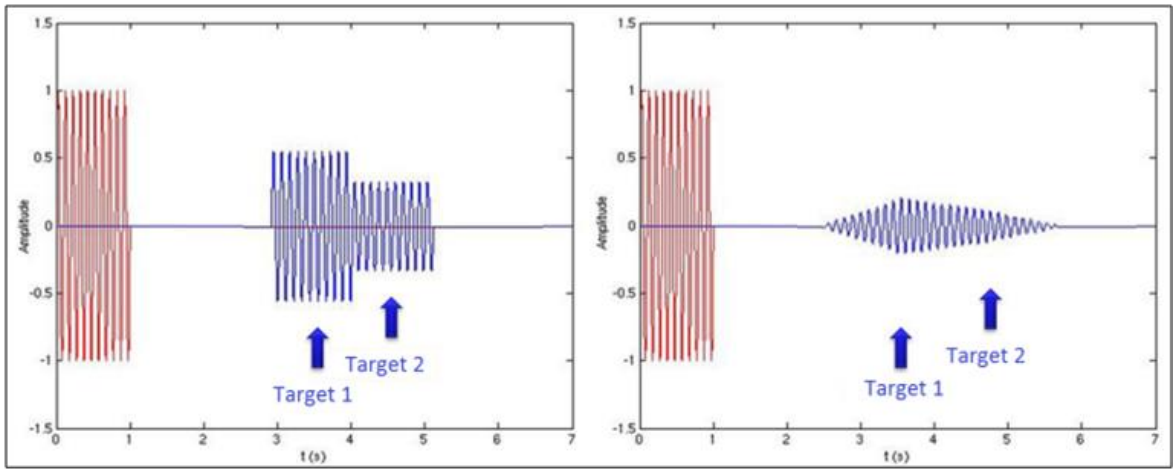
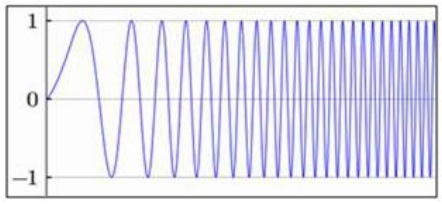
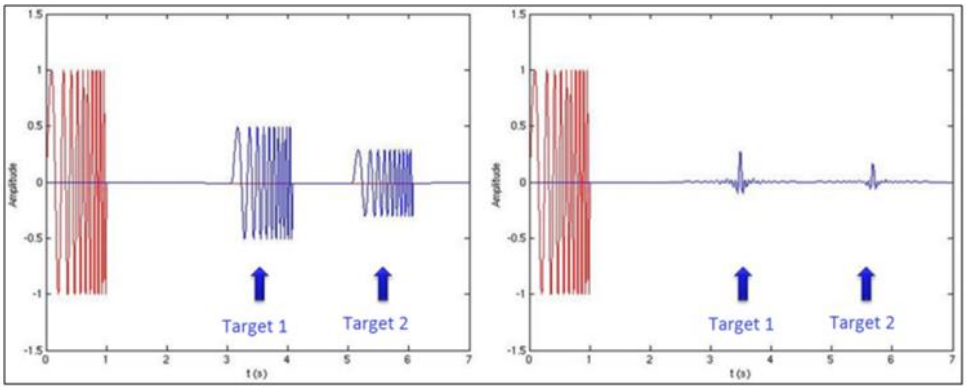


Figure 1. Detection of two closely targets using sinusoidal signal wave [1]



(a)



(b)

Figure 2. (a) Chirp signal, (b) detection of two closely targets using the pulse compression [1]

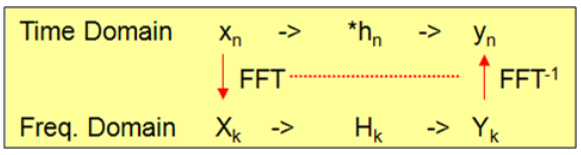


Figure 3. Cross-correlation between time and frequency [1]

2.2. C6678 DSP overview

The experimental platform consists of one development board EVM6678 as shown in Figure 4, which integrates one C6678 DSP and 512MB of DDR3 memory [25, 26]. The multi-core C6678 DSP provided by TI is a high-performance computing and low power system. It contains eight independent DSP cores, each core run at a frequency of 1GHz and has a peak performance of 16 GFLOPS for single precision floating point calculation. The C66x DSP core is based on a very long instruction word (VLIW) architecture. The instruction set also includes single input multiple data (SIMD) operating up to 128-bit vectors [4].

The DSP C6678 integrates three levels of memory. Each core has a 32-KB of level 1 for program (L1P) and 32-KB of level 1 for data (L1D). The level 1 is the nearest, and it is usually used as cache memory. In addition, each core has a local level 2 memory; it is slower than level 1, and its size is 512 KB. The level 3 or MSM is shared and is concurrently accessed by eight cores; its size is 4 MB. Furthermore, the eight DSP cores also access simultaneously to the external DDR memory.

For code development, the integrated development environment (IDE) code composer studio (CCS) has been used with C6000 compiler version v8.3.5. All optimization options provided by the compiler have been activated. The compiler also supports OpenMP 3.0, which allows rapid porting of existing multi-threaded codes to the multicore DSP. TI's C66x compiler translates the OpenMP into multi-threaded code with calls to a custom runtime library. The OpenMP framework was employed to instantiate individual threads across multiple cores. Pulse compression coefficients and input/output data have been allocated in MSM memory in order to be sahred between all cores, while L1 memory has been fully activated as cache.

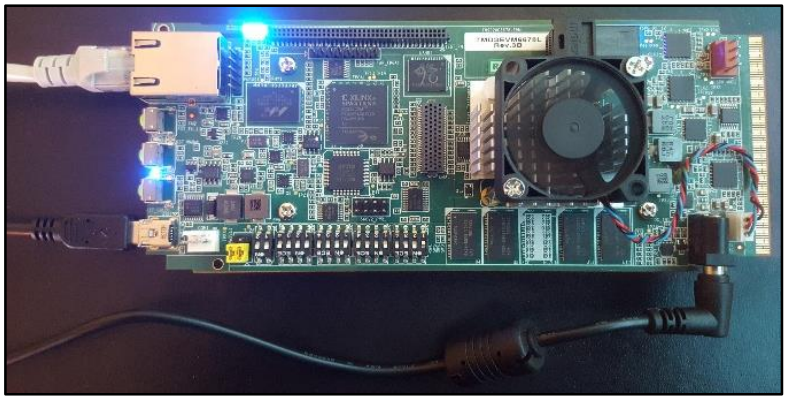


Figure 4. EVM6678 development board

2.3. Metrics for evaluating parallel processing performance

There are two metrics to evaluate performance of parallel processing: speedup (1) and parallel efficiency (2) [19]. An ideal parallel implementation leads to a speedup equal to the number of cores and to a parallel efficiency of 100%.

$$\text{speedup} = \frac{\text{Execution time of an application on 1 processor}}{\text{Execution time on P processors}} \quad (1)$$

$$\text{parallel efficiency} = \frac{\text{Speedup}}{\text{Number of cores}} * 100 \quad (2)$$

2.4. Proposed approach

The proposed approach aims to distribute the processing over the eight cores of the C6678 DSP. This approach is based on using MSM memory shared between all cores. We have placed pulse compression coefficients, input and output data in MSM memory in such a way that they are accessible to all cores at the same time. We have reserved seven memory boxes for synchronization; one box is dedicated for each core. Indeed, during the initialization phase, the master core (core 0) resets all these memory boxes and once arriving at the start of the parallel region, the master core set all boxes to one and begins processing its portion of data. Once the memory box of each core is set to one, the core starts processing its data portion. When ending its processing, the master core examines the states of the seven boxes and it would wait until it returns to state zero. This means that the other cores have also finished the processing. A diagram that illustrate the proposed method is presented in Figure 5.

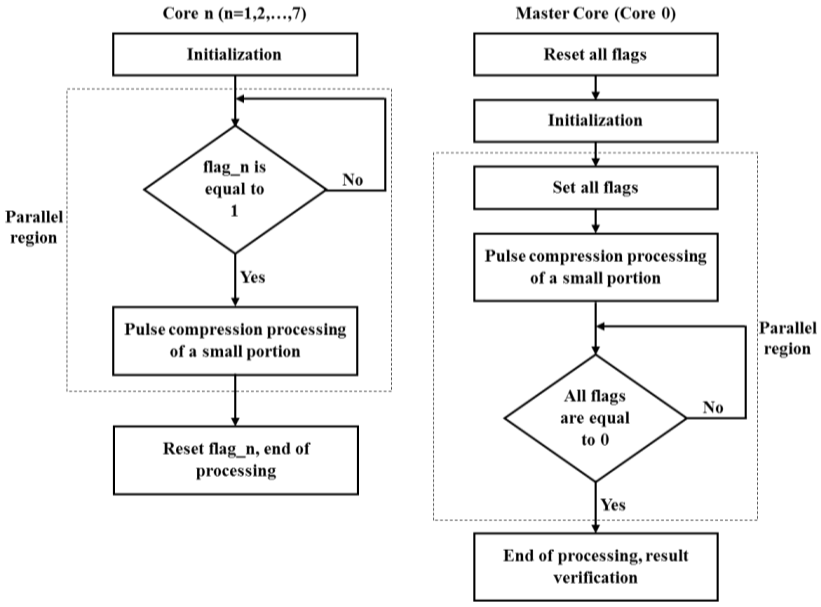


Figure 5. The proposed method

3. RESULTS AND DISCUSSION

3.1. Parallel implementation based on OpenMP

As described in section 2.1, pulse compression algorithm consists of three operations, FFT on input data, point-wise vector multiplication with pulse compression coefficients, and finally the IFFT to generate the output data. These three operations must be applied on all beams and pulses in case of pulse Doppler and GMTI applications, and on all pulses in case of SAR applications. In this work a use case of 256 iterations was chosen. Therefore, the software of the pulse compression consists of an external loop For, which repeats the three operations on all input data. OpenMP provides three scheduling techniques to control the manner in which loop iterations are distributed over the multiple cores. Thus, the scheduling method could have a major impact on performances. These methods are: static, guided and dynamic [19]. Experimental results are presented in Figure 6.

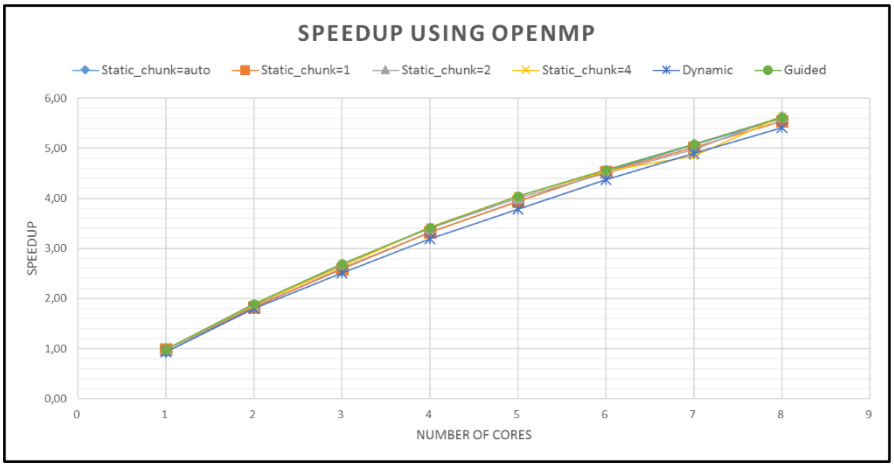


Figure 6. Parallel implementation results using OpenMP

From these results, it can be concluded that the three scheduling techniques give good performances. The speedup scales very well with the increase of the number of operational cores. When the eight cores are activated, the maximum speedup is equal to 5.6 with a corresponding parallel efficiency of 70%. This result can be explained by the overhead added by OpenMP framework to distribute data over the cores and to

synchronize the start and the end of a parallel region. Wang et al., [3] have obtained exactly the same result, however, Yu et al., [21] have obtained less value of the speedup that is equal in the best case to 1.

3.2. Parallel implementation using the proposed method

The proposed method presented in section 2.4 has been used to distribute the processing of pulse compression algorithm on multiple cores of the C6678 DSP. Experimental results are presented in Figure 7. Obtained results show that the speedup scales very well with the increase of the number of operational cores, with a small performance degradation in case where six and seven cores where activated. This depends on the number of iterations, which it is not a multiple of six and seven in our use case. A good choice of iterations number will lead to a best performance. When the eight cores are activated, the speedup achieves 7.5 with a corresponding parallel efficiency of 94%. Compared to Wang et al., [3] and to our previous research work [1], the proposed method gives the best performance.

Figure 8 presents a comparison between obtained results using the OpenMP framework and the proposed method. Thus, the proposed method leads to a gain of one core when the number of activated cores is equal to five and seven and a gain of two cores when the eight cores are activated. Therefore, our proposed method could be used as an alternative to OpenMP framework to distribute signal-processing algorithms over multi-core DSP. Radar applications are a good example.

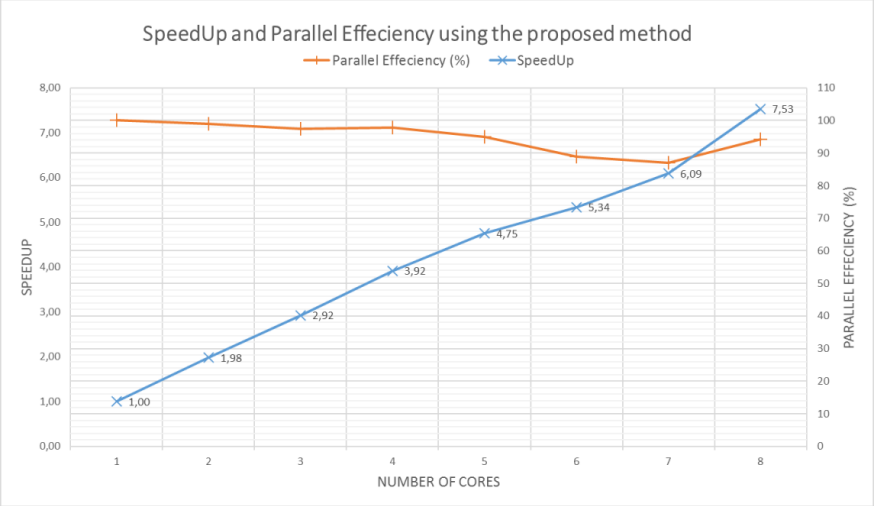


Figure 7. Parallel implementation results using the proposed method

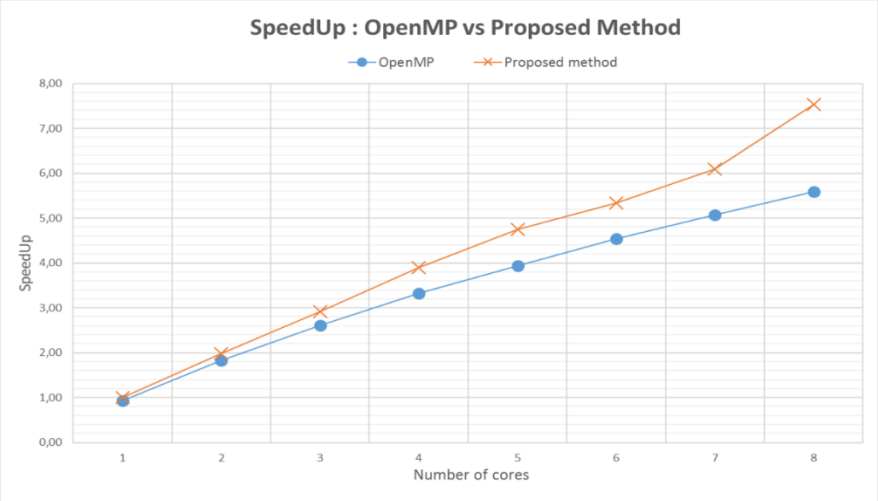


Figure 8. Parallel implementation result comparison

4. CONCLUSION

Pulse compression is the main processing step in several radar applications, such as pulse Doppler radar, GMTI and SAR. Its processing is based on cross-correlation. In order to optimize its processing, the cross-correlation was performed in frequency domain. We proposed the multi-core C6678 DSP as a real-time computing platform, which integrates eight independent cores with a shared memory. The goal of this paper was the evaluation of the OpenMP framework and the proposition of an optimized approach to distribute the processing over multiples cores. The proposed method consists of using shared memory to store synchronization flags, input and output data. Three scheduling techniques of OpenMP framework have been tested: static, guided and dynamic. These three techniques give the same performances with a maximum parallel efficiency of about 70% when the eight cores were activated. Obtained results using the proposed method lead to a speedup of about 7.5 and a parallel efficiency of about 94%, which is better than 70 % found in the previous works and obtained using OpenMP framework.

REFERENCES

[1] A. Klilou, et al., "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*, vol. 161, 2014.

[2] D. Bueno, et al., "Optimizing RapidIO Architectures for Onboard Processing," *ACM Transactions on Embedded Computing Systems*, vol. 9, no. 3, pp. 1-30, 2010.

[3] D. Wang and M. Ali, "Synthetic Aperture Radar on Low Power Multi-Core Digital Signal Processor," in *IEEE Conference on High Performance Extreme Computing (HPEC)*, Waltham, MA, 2012.

[4] Texas Instruments, "TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor," *Data Manual*, 2012.

[5] M. Najoui, et al., "VLIW DSP-Based Low-Level Instruction Scheme of Givens QR Decomposition for Real-Time Processing," *Journal of Circuits Systems and Computers*, vol. 26, no. 9, pp. 1-26, 2017.

[6] M. Bahtat, et al., "Instruction scheduling heuristic for an efficient FFT in VLIW processors with balanced resource usage," *Eurasip Journal on Advances in Signal Processing*, vol. 38, pp. 1-21, 2016.

[7] R. Berg, et al., "Highly efficient image registration for embedded systems using a distributed multicore DSP architecture," *Journal of Real-Time Image Processing*, vol. 14, no. 2, pp. 341-361, 2018.

[8] N. Bahri, et al., "Real-time H264/AVC High Definition video encoder on a Multicore DSP TMS320C6678," *in International Conference on Computer Vision and Image Analysis Applications*, 2015.

[9] A. Klilou, et al., "Real-time parallel implementation of road traffic radar video processing algorithms on a parallel architecture based on DSP and ARM processors," in *2015 15th International Conference on Intelligent Systems Design and Applications*, pp. 183-188, 2015.

[10] A. E. Abdelkareem, et al., "Design and implementation of an embedded system for software defined radio," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 6, pp. 3484-3491, 2017.

[11] A. Arsalane, et al., "The assessment of fresh and spoiled beef meat using a prototype device based on GigE Vision camera and DSP," *Journal of Food Measurement and Characterization*, vol. 13, no. 3, pp. 1730-1738, 2019.

[12] A. Arsalane, et al., "Beef and horse meat discrimination and storage time classification using a portable device based on DSP and PCA method," *International Journal of Intelligent Enterprise*, vol. 4, no. 1-2, pp. 58-75, 2017.

[13] A. Arsalane, et al., "Building a portable device based on DSP for meat discrimination," in *Proceedings 2016 International Conference on Engineering and MIS (ICEMIS 2016)*, 2016.

[14] A. Arsalane, et al., "An embedded system based on DSP platform and PCA-SVM algorithms for rapid beef meat freshness prediction and identification," *Computers and Electronics in Agriculture*, vol. 152, pp. 385-392, 2018.

[15] A. Arsalane, et al., "Artificial vision and embedded systems as alternative tools for evaluating beef meat freshness," in *the 6th International Conference on Optimization and Applications*, Beni Mellal, Morocco, 2020.

[16] A. Klilou, et al., "Case studies of data traffic management on a high-performance computing system based on multi-DSPs and Serial RapidIO interconnect," in *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, pp. 1-6, 2016.

[17] A. Klilou, et al., "Performance optimization of high-speed Interconnect Serial RapidIO for onboard processing," in *2012 International Conference on Complex Systems (ICCS)*, pp. 1-6, 2012.

[18] L. Huang, et al., "Parallelizing Ultrasound Image Processing using OpenMP on Multicore Embedded Systems," in *2012 IEEE Global High Tech Congress on Electronics (Ghtce)*, 2012.

[19] B. Chapman, et al., "Using OpenMP Portable Shared Memory Parallel Programming," *The MIT Press*, 2007.

[20] R. Mego and T. Fryza, "Performance of Parallel Algorithms Using OpenMP," *2013 23rd International Conference Radioelektronika (Radioelektronika)*, pp. 236-239, 2013.

[21] X. N. Yu, et al., "An Implementation of Real-Time Phased Array Radar Fundamental Functions on a DSP-Focused, High-Performance, Embedded Computing Platform," *Aerospace*, vol. 3, no. 3, pp. 28-50, 2016.

[22] D. C. Schleher, "MTI and Pulsed Doppler Radar," *Artech House Publishers*, 1991.

[23] M. Z. Hussain and K. N. Parvin, "Low power and high performance FFT with different radices," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 8, no. 2, pp. 99-106, 2019.

[24] A. Manimaran and A. K. Thomas, "Design of "32" point split radix based multipath delay commutator FFT architecture for low power applications," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 11, no. 3, pp. 1042-1047, 2018.
[25] Advantech, Texas Instruments, "TMDSEVM6678L EVM Technical Reference Manual, Version 2.01," *Advantech*, 2012.
[26] A. Kharin, et al., "Teaching multi-core DSP implementation on EVM C6678 board," in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017.

BIOGRAPHIES OF AUTHORS

Abdessamad Klilou was born in Marrakech, Morocco, in 1987. He received an engineer's degree in 2010 and a Ph.D degree in 2016 from the University of Cady Ayyad, Marrakech, Morocco. Since 2017, he is a professor at the department of electrical engineering in the Faculty of Sciences and Technology, University of Sultan Moulay Slimane, Beni Mellal Morocco. His area of research is focused on parallel and real time optimization of signal processing algorithms on multi-core and multi-processors parallel machine.

Assia Arsalane received an engineer's degree in Electrical Engineering from the National School of Applied Sciences of Khouribga in 2014 and a Ph.D degree in 2019 from the University of Hassan I, Settat, Morocco. Since 2018, she is a visiting professor in the department of mechatronics, High School of Technologies, University of Sultan Moulay Slimane, Beni Mellal Morocco. Her area of research includes artificial intelligence, machine vision, image processing and embedded systems.