Synthetic-Aperture Radar Robust Reconfigurable Optimized Computing Architecture

Helena Cruz, José Monteiro, Horácio Neto & Rui Policarpo Duarte
INESC-ID/IST-UL, Lisboa
helena.cruz@tecnico.ulisboa.pt, hcn@inesc-id.pt, jcm@inesc-id.pt & rui.duarte@tecnico.ulisboa.

Synthetic-Aperture Radar

Remote sensing technologies such as Synthetic-Aperture Radar (SAR) have been widely used **monitor** the surface of the Earth, in particular:

- ships and oil spills;
- drought and landslides;
- deforestation and fires;
- hurricanes, volcano eruptions and earthquakes.

SAR sensors can be mounted on-board flying platforms such as **satellites**, **aircrafts** and **drones**. The main strength of SAR is that it operates even in the presence of clouds, smoke and rain and without a light source.

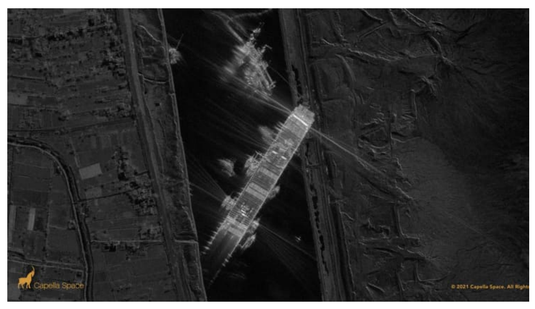


Figure 1: SAR image of the Evergreen ship stuck on the Suez Canal, provided by Capella Space.

Moreover, with the advancements in the technology and signal processing methods, there are increasing business opportunities for satellites and drones equipped with **lightweight**, **small**, and **autonomous systems** for **on-board processing** and generation of SAR images and subsequent broadcasting them, avoiding the time-consuming processing data at the receivers.

SAR Image Generation Algorithms

There are many SAR image generations algorithms, and they can be divided into two groups: **FFT-based** and **non-FFT-based**.

FFT-based algorithms are often more efficient, however, they introduce **warping** and distortion in the final image, especially when the pixels are far away from the image center.

Non-FFT-based, such as the Backprojection algorithm, have a higher computational complexity, however, the quality of the images is **superior**.

The Backprojection algorithm is, for this reason, chosen to implement a fault-tolerant and optimized architecture for SAR imaging.

The principle of the Backprojection algorithm is that **every** pulse has a contribution for **every** pixel of the image. For every pulse, the contribution is calculated and accumulated in the pixel value. These calculations are **independent** and can be performed in **parallel**.

Implementation of the Backprojection Algorithm

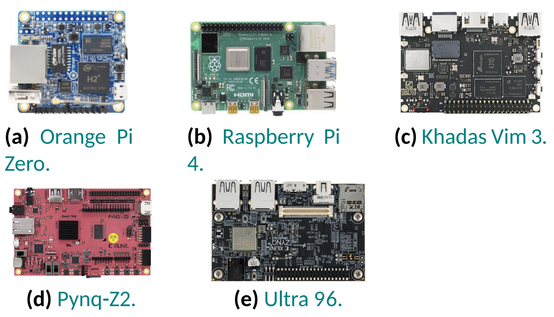
Devices used for on-board processing must respect the following requisites:

- Size;
- Weight;
- Power consumption;

These requisites are also known as **SWaP**. The platforms used to implement the Backprojection algorithm are the following:

- Pynq-Z2;
- Ultra96;
- Orange Pi Zero;
- Raspberry Pi B;
- Khadas Vim 3.

There are two different implementations of the Backprojection: **single-core** and **multi-core**. The costly operations of the algorithm, sine and cosine, as seen in the profiling section, were also implemented in two different ways: **floating-point** and **fixed-point**. The execution time of each implementation and respective power consumption of the board is in Table 1. For each board, the most **efficient** implementation is highlighted.



(a) Orange Pi Zero. **(b)** Raspberry Pi 4. **(c)** Khadas Vim 3. **(d)** Pynq-Z2. **(e)** Ultra 96.

Figure 2: Single board computers, in the first row, and Xilinx SoC-FPGA boards, in the bottom row, used in the Backprojection algorithm implementation.

Table 1: Performance and power consumption according to device and implementation.

Board	sin/cos	core	t [s]	p [mW]	e [mWh]
Pynq-Z2	float	single-core	473	1715	225
	fixed-point	single-core	125	1720	60
	float	multi-core	238	1840	122
	fixed-point	multi-core	63	1810	32
Ultra96	float	single-core	273	2035	154
	fixed-point	single-core	124	2020	69
	float	multi-core	69	2205	42
	fixed-point	multi-core	31	2130	18
Orange Pi Zero	float	single	154	1210	52
	fixed	single	174	1210	58
	float	multi	38	2150	23
	fixed	multi	44	2075	26
Raspberry Pi B	float	single	57	3397	54
	fixed	single	73	3381	68
	float	multi	28	3652	28
	fixed	multi	38	3631	38
Khadas Vim 3	float	single	47	2660	35
	fixed	single	43	2720	32
	float	multi	11	5660	17
	fixed	multi	12	5610	19

The most **efficient** board is the Khadas Vim 3, with 17 mWh. A close second is the Ultra96, with 18 mWh. This is a software implementation, which means the FPGA fabric of the Ultra96 was not used. With an **accelerator**, the Backprojection algorithm implementation can be even more efficient, highlighting the potential of hardware accelerators.

Backprojection Algorithm Profiling

- The **profiling** of the BP algorithm running on a single core of the ARM A9 processor of the target Zynq device was required to determine which parts of the algorithm should be accelerated.
- The implementation of the BP algorithm adopted is available in [1].
- The best option for the acceleration are the **sine** and **cosine** functions.

Operation	Time [ns]	Execution Time [%]
Sqrt	50	1.3
Sin+Cos	3108	84.3
Misc	530	14.4
Total	3688	100.0

Table 2: Execution times for the operations in the implementation of BP.

SAR Backprojection Accelerator

An **accelerator** was developed for the Backprojection algorithm, targeting the most time consuming operations and was specified using **Xilinx HLS**.

- Using HLS and maintaining the floating-point representation allows to reutilize parts of the source code and guarantees that the result is the same as the original implementation.
- The accelerator was implemented as a single IP core, where it receives the range values and samples for 512 pulses.

Resource	Utilization	Total on Zynq-7020 [%]
BRAM18K	2	1.0%
DSP48E	34	15.0%
LUTs	13986	26.0%

Table 3: Estimate of resources required to implement the BP accelerator reported by Vivado HLS.

´The range values are double precision floating-point values whereas the samples are complex single-precision floating-point values.

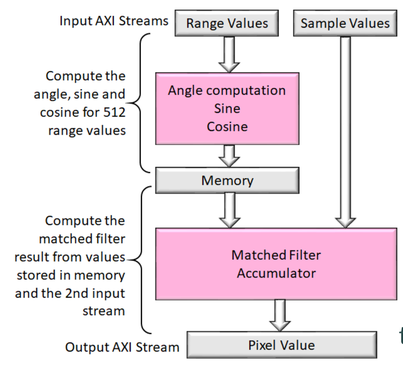


Figure 3: Organization of the accelerator.

The accelerator is organized as follows:

- The first loop obtains the input data for the range values from the streaming interface, computes their product to serve as input to trigonometric functions and stores the result in local memories.
- The second loop receives the pulse samples also via the streaming interface, performs the complex multiplication and writes the result to the output streaming interface.

Figure 3 illustrates the sequence diagram of the relations between the building blocks of the accelerator.

Table 3 summarizes the **FPGA resources** required to implement the BP accelerator from the specification. The HLS tool produced a circuit design capable of operating at **100 MHz**, resulting in an IP core which requires a minimum of 60 clock cycles in latency, of which 24 cycles are required by the CORDIC IP.

- The accelerator was tested on a **Zynq-7020** device installed on a Pynq-Z2;
- The processing times for the computations made by the accelerator in software required 1667.3 us, whereas the same computations in the accelerator required only 37.31 us, a reduction of $44.68\times$;
- The processing time for a 512x512 image was $7.7\times$ faster with the accelerator;
- The **average power consumption** of the whole system is 1.796 W.

Future Work

- The accelerator used floating-point units, which guarantee the same results but are slower than fixed-point units. An accelerator for the Backprojection algorithm in **fixed-point** is currently under development.
- **Reducing data transmission** and increasing the number of operations performed in the accelerator and as much as possible contributes to a more **efficient** implementation.
- Embedded systems are subject to **faults**, especially those in space which are under the effects of **radiation**. Fault tolerance mechanisms are necessary to ensure the proper functioning of the system. **Traditional fault tolerance mechanisms use redundancy and are algorithm-agnostic.** A methodology to develop fault tolerance mechanisms **customized to the algorithm** will be studied. This methodology takes into consideration the fact that not all variables have the same impact into the final image and uses that when protecting the algorithm, using **reduced-precision redundancy** and **approximations**, thus minimizing the overhead of the mechanism in the system.

References

[1] Kevin Barker, Thomas Benson, Dan Campbell, David Ediger, Roberto Gioiosa, Adolfy Hoisie, Darren Kerbyson, Joseph Manzano, Andres Marquez, Leon Song, Nathan Tallent, and Antonino Tumeo. *PERFECT Benchmark Suite Manual*. Pacific Northwest National Laboratory and Georgia Tech Research Institute, December 2013.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT), under grants with references UIDB/50021/2020 (INESC-ID multi-annual funding) and SARRROCA (PTDC/EEI-HAC/31819/2017). Helena Cruz would like to acknowledge Fundação para a Ciência e a Tecnologia for the support through grant SFRH/BD/144133/2019.