High-Level Synthesis-Based On-board Payload Data Processing considering the Roofline Model

Seungah Lee, Ruben Salvador, Angeliki Kritikakou, Olivier Sentieys, Julien Galizzi, Emmanuel Casseau

High-Level Synthesis-Based On-board Payload Data Processing considering the Roofline Model

Seungah Lee
Univ Rennes, Inria, CNRS, IRISA
France

Ruben Salvador
CentraleSupélec, Inria, Univ Rennes, CNRS, IRISA
France

Angeliki Kritikakou
Univ Rennes, Inria, CNRS, IRISA
France

Olivier Sentieys
Univ Rennes, Inria, CNRS, IRISA
France

Julien Galizzi
CNES
France

Emmanuel Casseau
Univ Rennes, Inria, CNRS, IRISA
France

Abstract

On-board payload data processing can be performed by developing space-qualified heterogeneous Multiprocessor System-on-Chips (MPSoCs). We present key compute-intensive payload algorithms, based on a survey with space science researchers, including the two-dimensional Fast Fourier Transform (2-D FFT). Also, we propose to perform design space exploration by combining the roofline performance model with High-Level Synthesis (HLS) for hardware accelerator architecture design. The roofline model visualizes the limits of a given architecture regarding Input/Output (I/O) bandwidth and computational performance, along with the achieved performance for different implementations. HLS is an interesting option in developing FPGA-based on-board processing applications for payload teams that need to adjust architecture specifications through design reviews and have limited expertise in Hardware Description Languages (HDLs). In this paper, we focus on an FPGA-based MPSoC thanks to recently released radiation-hardened heterogeneous embedded platforms.

Index Terms

On-board processing, FFT, System-on-Chip (SoC), FPGA, High-level synthesis (HLS), Roofline performance model

I. Introduction

Satellites embark payload sensors that require data processing to obtain data products so that payload data processing designates a series of processing for a payload. Payload data processing has traditionally been performed on the ground, after the downlink of raw data, due to the lack of high-performance processors tolerant to radiation [1]. However, with recent space-qualified heterogeneous Multiprocessor System-on-Chips (MPSoCs), high-performance on-board processing is expected. Migrating ground-based data processing pipelines to on-board systems can significantly increase system autonomy and decrease monitoring and detection latency.

The purpose of this work is to explore the possibility of such migration by leveraging the recent heterogeneous accelerator architectures integrating a Field-Programmable Gate Array (FPGA) with Arm processors for both scientific missions and NewSpace, such as space exploration and Earth observation. To achieve that, the first step is to identify the most important candidates for payload data processing to be moved on board. More precisely, we perform a survey on on-board embedded systems (Table I) and payload processing algorithms regarding recent and near-future missions, in which European research institutes are involved. Based on this survey, we selected as a case study the most recurring payload algorithm found, i.e., the Fast Fourier Transform (FFT).

Most current missions with FFT on-board implementations focus on the one-dimensional (1-D) FFT. There are few cases of 2-D FFT implementations on space-qualified processors, which, however, have low performance with long latency. One reason is that the processor, usually paired with small on-chip memories, requires a significant amount of external memory access. To address this limitation, application-specific hardware accelerators can be used for compute-intensive algorithms, including the 2-D FFT. However, such implementations should efficiently meet the restricted on-board resources and maintain design productivity close to software-only implementations. Therefore, we propose a Design Space Exploration (DSE) approach, where we combine the roofline performance model [8] with High-Level Synthesis (HLS)-based hardware design. On the one hand, the roofline performance model allows us to bound accelerator architectures considering memory access, application, and architecture constraints [9]. On the other hand, HLS enables the exploration of various hardware designs effectively. Specifications using the C++ language, combined with HLS, make hardware design less complex compared to using a Hardware Description Language (HDL), as high-level descriptions can be interpreted to create digital hardware with the same functionality.

This research was supported by CNES and University of Rennes thanks to PhD funding granted to Seungah Lee.

TABLE I
PAYLOAD DATA PROCESSING HARDWARE EXAMPLES USED FOR THE LAST 10 YEARS AND NEAR-FUTURE MISSIONS

Mission	Launch year	Payload data processing hardware	Ref.
PLATO	2026*	GR712RC LEON3FT ASIC, MDPA LEON2FT ASIC, LEON3FT with RTAX2000 FPGA	[2]
JUICE	2023	GR712RC LEON3FT ASIC	[3]
Solar Orbiter	2020	LEON3FT with RTAX4000 FPGA	[4]
CHEOPS	2019	GR712RC LEON3FT ASIC	[5]
BepiColombo	2018	HIREC-MIPS-HR5000 CPU, RTAX2000 FPGA	[6]
Gaia	2013	Custom pre-processing board, SCS750 PowerPC board	[7]

*Estimated launch schedule

TABLE II
PRIORITIZED ALGORITHMS FROM THE SURVEY WITH PAYLOAD TEAMS

Classification	Sub-classification	Number of users
Fourier transform	FFT, IFFT, DFT	5
Filter	IIR	4
	CIC	4
	Kalman	1
AI/ML		4
Compression	CCSDS 121	3
	CCSDS 122	3
	CCSDS 123	3
	CCSDS 124	3
Optimization	Interpolation	3
	Fitting and correlation	2
	Gradient descent	2
Histogram		1
Digital Elevation Model		1

Considering recently developed hardware architectures, we propose using Commercial Off-The-Shelf (COTS) heterogeneous embedded systems. Specifically, we use the Xilinx Zynq UltraScale+ MPSoC as an initial target after reviewing the specifications of radiation-hardened heterogeneous embedded platforms recently released, e.g., NG-Ultra and Versal [10], [11].

II. ON-BOARD DATA PROCESSING ALGORITHMS FOR SCIENTIFIC MISSIONS AND NEWSPACE: THE FFT CASE

We surveyed current on-board and ground processing algorithms to identify the trends of on-board data processing and the expected needs of future missions. Besides, in collaboration with scientific payload teams, we identified and prioritized compute-intensive algorithms applicable to several types of payloads, depicted in Table II. These algorithms are challenging and thus hardly executed on an existing radiation-hardened On-Board Computer (OBC) with flight heritage. According to our survey, the 2-D FFT is widely used in diverse applications, including Synthesis Aperture Radar (SAR) [12] and space telescopes [13]. Therefore, we selected the 2-D FFT as a case study based on the computational requirements of payloads, such as image size and data accuracy. The 2-D FFT can be a demanding algorithm due to its column-wise processing step or transpose, which can easily become a computational and memory bottleneck [14].

The Fourier transform is a transformation converting a time-domain function $f(x)$ to a frequency-domain function $\hat{f}(\xi)$ and its general expression can be described using the infinite integral of integration given by:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi\xi x} \, \mathrm{d}x \tag{1}$$

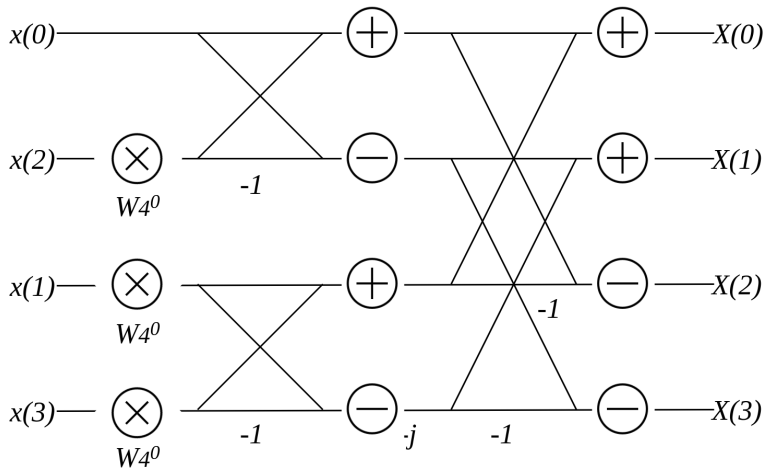

Fig. 1. Butterfly of the DIT radix-4 FFT

The Discrete Fourier Transform (DFT) is the Fourier transform of a finite number of discrete signals, which is generally used in digital signal processing. The N points of time-domain complex-number signals $x(n)$, where $n = 0, 1, \ldots, N-1$, are transformed into the results of the DFT $X(k)$ on the frequency k where $k = 0, 1, \ldots, N-1$.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}} \tag{2}$$

Similarly, for an $N \times N$ matrix of signals $x(n_1, n_2)$ where $n_1, n_2 = 0, 1, \ldots, N-1$, the 2-D DFT results $X(k_1, k_2)$ are defined as below regarding the frequency k_1 and k_2, where $k_1, k_2 = 0, 1, \ldots, N-1$.

$$X(k_1, k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \cdot e^{\frac{-j2\pi(k_1 n_1 + k_2 n_2)}{N}} \tag{3}$$

From the above equations, we can derive that the complexity of DFT is $\mathcal{O}(N^2)$ for the 1-D DFT and $\mathcal{O}(N^3)$ for the 2-D DFT.

As the DFT computation is slow, there have been several attempts to increase DFT performance: this is how the FFT was devised. The FFT is an algorithm for calculating the DFT efficiently by reducing the computational complexity. The Cooley-Tukey algorithm is the most popular FFT algorithm, which breaks down the summation recursively into a certain number called radix [15]. The radix determines the number of arithmetic operations of FFT which is consequently relevant to the computational cost of FFT. In this paper, we focus on the radix-4 FFT considering the library implemented in section IV as a use case. For instance, we deal with the N-point Decimation-In-Time (DIT) radix-4 FFT which separates the summation notation of (2) into 4 different terms ($n = 4m, 4m+1, 4m+2, 4m+3$) where the upper limit of summations is $\frac{N}{4} - 1$. The summation can be represented as follows:

$$\begin{aligned} X(k) = \sum_{m=0}^{\frac{N}{4}-1} x(4m) e^{\frac{-j2\pi k(4m)}{N}} + \sum_{m=0}^{\frac{N}{4}-1} x(4m+1) e^{\frac{-j2\pi k(4m+1)}{N}} \\ + \sum_{m=0}^{\frac{N}{4}-1} x(4m+2) e^{\frac{-j2\pi k(4m+2)}{N}} + \sum_{m=0}^{\frac{N}{4}-1} x(4m+3) e^{\frac{-j2\pi k(4m+3)}{N}} \end{aligned} \tag{4}$$

Considering the symmetry of the exponential factors, we can simplify the exponential terms into twiddle factors $W_N^k = e^{\frac{-j2\pi k}{N}}$ which are represented in the butterfly operations in the context of the Cooley–Tukey FFT algorithm. The butterfly operation is a unit of the FFT and the total number of butterflies of radix-R FFT is the multiplication of the number of stages ($log_R N$) and the number of input data divided by radix ($\frac{N}{R}$), thus the total number of butterflies is $\frac{N}{R} log_R N$. Based on this definition, each radix-4 FFT butterfly has 3 complex multiplications and 8 complex additions as shown in Fig. 1. For the radix-4 FFT, the total number of complex multiplications is $3\frac{N}{4} log_4 N$, and the total number of complex additions/subtractions is $2N log_4 N$. If we want to use the hardware composed of only real operators, such as FPGAs, understanding real arithmetic operations is also necessary, as the hardware cannot perform complex operations directly. The total number of real multiplications is $3N log_4 N$ and the number of real additions/subtractions is $\frac{11}{2} N log_4 N$ [16]. Based on the required number of real operations, we

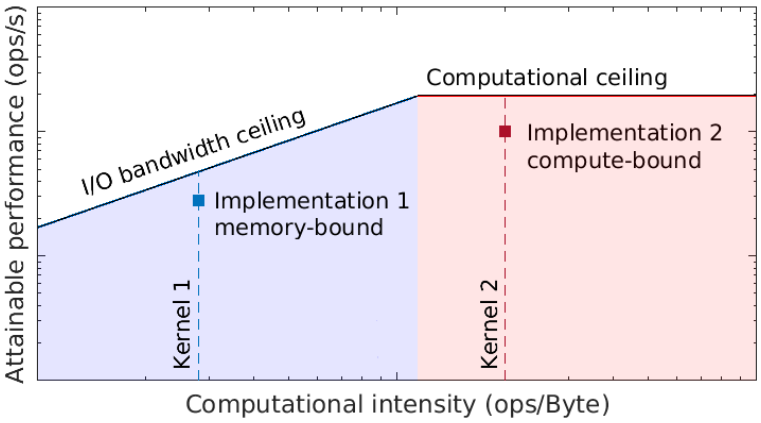

Fig. 2. Roofline model [18]

can derive the computational complexity. For N-point data, the computational complexity of the DFT $\mathcal{O}(N^2)$ is reduced to $\mathcal{O}(Nlog_RN)$ by using the radix-R FFT. If we consider a 2-D $N \times N$ matrix, we first perform the row-wise 1-D FFT of the given matrix by repeating the N-point FFT for the number of rows which is N times in this case. So, the computational complexity becomes $\mathcal{O}(N^2log_RN)$ because the complexity of 1-D FFT is multiplied by N. Then, we conduct the column-wise 1-D FFT of the matrix, whose computational complexity is still $\mathcal{O}(N^2log_RN)$ because a constant factor does not affect big O notation. The complexity and the number of real operations for both 1-D FFT and 2-D FFT on radix-4 are represented in Table III which are used in section IV to compute the characteristics of 2-D kernel design.

III. DSE Methodology

DSE helps to navigate the space of possible design solutions by formulating the design problem as a multi-objective optimization problem, which considers latency, area, and power [17]. By associating the roofline model with designs achieved from HLS, we explore several hardware accelerator designs on an FPGA and analyze their suitability for scientific missions and NewSpace.

A. Roofline model

The roofline model characterizes designs using the limits of bandwidth and performance on a given architecture [8]. Even though the original roofline model is appropriate to von Neumann architectures, namely multicore Central Processing Unit (CPU) architectures and Graphics Processing Unit (GPU), the model has been extended for FPGAs considering their reconfigurable characteristics [18].

The FPGA roofline model, drawn as a log-log scale in Fig.2, is based on two ceilings that present the limits of a given architecture: i) Input/Output (I/O) bandwidth ceiling and ii) computational ceiling. Consequently, any application implementation is bounded either by the I/O bandwidth ceiling (memory-bound) or the computational ceiling (compute-bound). The computational ceiling is calculated using the total number of specific computational resources, the number of required resources per arithmetic operation, and the maximum clock frequency. If an FPGA reads data from external memory to perform processing, the I/O bandwidth ceiling depends on the external memory and its relevant interface. In other words, the one with smaller bandwidth becomes the I/O bandwidth of a given system. For external memory, its bandwidth is the multiplication of memory data rate and memory channel bit-width. In the case of an interface, the bandwidth is the multiplication of clock frequency, transfer data bit-width, and the number of interface ports.

We can also indicate the implementations of a kernel, corresponding to each point in Fig. 2, to see whether any ceilings bound the design. In this context, the kernel defines a routine running on a hardware accelerator such as an FPGA, which is

TABLE III
COMPLEXITY, REAL ADDITIONS, AND REAL MULTIPLICATIONS OF THE RADIX-4 FFT

	1-D FFT	2-D FFT
Complexity	$\mathcal{O}(Nlog_4N)$	$\mathcal{O}(N^2log_4N)$
Real additions/subtractions	$\frac{11}{2}Nlog_4N$	$11N^2log_4N$
Real multiplications	$3Nlog_4N$	$6N^2log_4N$

distinguished from an application executed on a CPU. The x-axis value of an implementation is the *computational intensity*, which is the number of total operations divided by the number of total data bytes accessed, and the y-axis value is the *performance*, represented as the number of operations divided by execution time.

B. HLS-based hardware accelerators

We use HLS based on C/C++ instead of using traditional HDLs for the design of hardware accelerators. Comparatively, HLS enables the simpler generation of different designs meeting the application's constraints faster and easier. To generate efficient hardware accelerators, HLS codes must be optimized accordingly. This requires refactoring an initial, functional C/C++ code carefully and using appropriate *pragmas and directives* to help the compiler generate hardware that efficiently uses the available memory and computational resources of FPGA for a considered application [17], [19]. Overall, HLS is both efficient and promising, simplifying the development not only of hardware but also of software-hardware co-design. Each HLS tool provides a large list of pragmas and directives that help users to perform the fine-tuning of their architecture. In this paper, the HLS tool we use is Vitis HLS from AMD-Xilinx. We present frequently used pragmas and directives below, which are later used in section IV-D [20], [21]: loop unrolling, loop pipelining, task pipelining, and array partitioning.

Loop unrolling unrolls a loop and executes all or some iterations of one loop in parallel (full unroll or partial unroll). This pragma decreases intervals and latency by enhancing parallelism, whereas it increases resource utilization. The interval is the number of clock cycles between two loop iterations in a row. Vitis HLS provides `#pragma HLS unroll`, which allows full unroll without any syntax option and enables partial unroll with the option factor=N, where N is the number of loop bodies replicated.

Loop pipelining enables the concurrent execution of operations at the cycle level so that it decreases the initiation interval (II) for a loop. The II indicates the number of clock cycles spent before starting the next loop iteration [22]. Thus an II=1 means a new loop iteration is started every clock cycle. This reduces intervals and latency, increasing the minimum amount of resources. Vitis HLS has `#pragma HLS pipeline` which tries the minimum II (i.e., 1) unless the syntax option II=N is statically specified for the number of II.

Task pipelining allows functions and loops to perform concurrent executions of coarse-grained operations taking a number of clock cycles, improving the general throughput of the architecture. Unlike the loop pipelining dealing with cycle-level pipelines, the behavior of relevant functions and loops cannot be statically scheduled. Users can utilize `#pragma HLS dataflow` for task-level pipelining in Vitis HLS.

Array partitioning subdivides an array into several smaller arrays. Arrays are stored on FPGA block RAMs (BRAM), which have single or dual ports. If a user implements only one large memory and needs to read and write multiple times, the small number of ports can drop throughput. However, this problem can be solved by partitioning a large memory into smaller memories or registers: the increased number of ports can then enhance throughput. `#pragma HLS array_partition` in Vitis HLS is defined with an array variable name and a partition type.

As a result, combining the roofline performance model and HLS-based design techniques enables developers to rapidly generate and evaluate processing architectures with different performance/resource trade-offs, reducing the time to prototype implementations [9]. Such an approach is suitable for payload teams where instrumentation specifications change depending on the mission and design review and where human resources are limited for developing on-board processing applications using an HDL.

IV. DSE use-case: the 2-D FFT

We present the FPGA roofline model and relevant DSE based on Vitis HLS targeting the Xilinx Zynq UltraScale+ ZCU102 evaluation board and the 2-D FFT as a use case.

A. Algorithm requirements

We selected the 2-D FFT considering its necessity in the space science community (Table II). More precisely, we selected the SVOM ECLAIRs code-masked telescope to extract technical requirements. The ECLAIRs telescope embarks a LEON3-based on-board payload control unit, processing the image trigger algorithm every 20.48 sec. The algorithm includes the deconvolution of 200×200-sized images based on the 2-D FFT which takes around 2 sec per image on board [23], [24]. Assuming that the 2-D FFT per image accounts for at least 5% of execution time, we set 100 ms as the time requirement in this paper. For data precision, we use single-precision floating-point data and 27-bit fixed-point data. The floating-point arithmetic operations are frequently used in astronomy as it is suitable for high-accuracy data and software implementation. However, not every Digital Signal Processing (DSP) block of FPGA implements floating-point support naturally. In other words, multiple DSP blocks can be required to perform floating-point computations depending on the FPGA model. For fixed-point computations, we use the maximum size of fixed-point data whose arithmetic operation fits in one of the DSP blocks available in the device contained in the board used in this paper, so we can balance resource utilization and acceptable data bit-width.

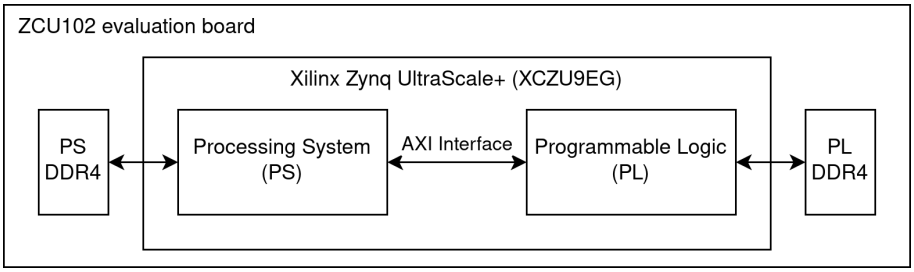

Fig. 3. ZCU102 evaluation board configuration

B. FPGA platform and HLS

The 2-D FFT hardware accelerators are designed using Vitis HLS 2021.2. The ZCU102 is used as an evaluation board featuring a Xilinx Zynq UltraScale+ XCZU9EG device. The Zynq UltraScale+ is a heterogeneous MPSoC that includes a quad-core Arm Cortex-A53 and a dual-core Arm Cortex-R5 called Processing System (PS), and an embedded FPGA called Programmable Logic (PL). The PL contains 600K System Logic Cells, 2,520 DSP blocks, 32.1 Mb BRAM, and 548K Flip-Flops (FF). The DSP blocks of Zynq UltraScale+ (DSP48E2) are not floating-point DSP blocks, so some extra hardware implemented in regular logic is needed for operations like decimal point alignment, among others. Each DSP block comprises a 27-bit$\times$18-bit input data multiplier and a 48-bit accumulator. In the ZCU102, the Zynq UltraScale+ is connected to external DDR4 memories (PS DDR4, PL DDR4), and the PS and PL are connected via an Advanced eXtensible Interface (AXI) as presented in Fig. 3. The bandwidth of the DDR memories and the AXI interface are considered to calculate the theoretical peak I/O bandwidth.

C. Theoretical FPGA roofline model

Fig. 4 depicts the obtained FPGA roofline model considering the hardware specifications of the ZCU102 board. As complex FFT computation on an FPGA requires real multiplications and real additions, we only consider DSP blocks as a computational resource type in this study [25].

First, the theoretical maximum computational ceiling is based on the total number of DSP blocks and their maximum frequency [26], assuming a DSP block handles one operation (op). The theoretical computational ceiling C_{th} is:

$$\begin{aligned} C_{th} &= 2520 \text{ DSP blocks} \times 775 \text{ MHz} \times 1 \text{ op/DSP block} \\ &= 1953 \text{ Gops/sec} \end{aligned} \tag{5}$$

FPGAs are reconfigurable platforms that can implement different custom processing architectures. As a result, the ceilings in FPGA rooflines are not unique when one considers realistic occupations achievable by synthesis tools (which do not reach

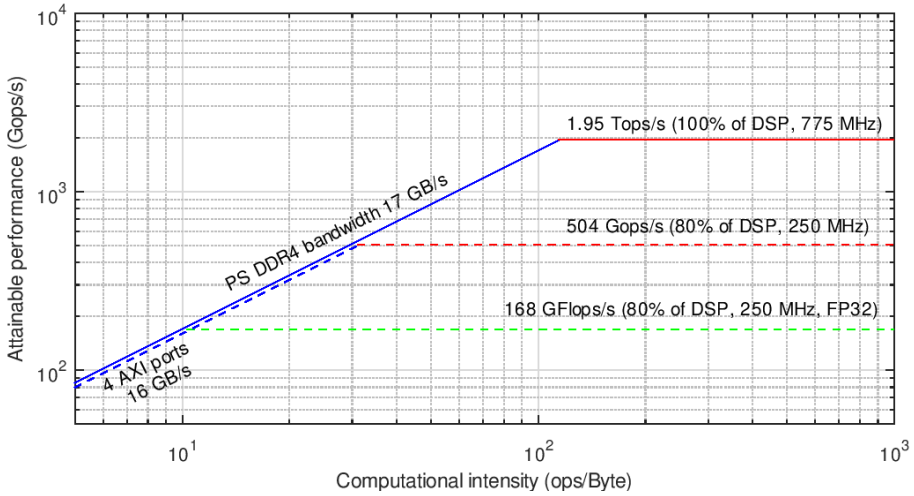

Fig. 4. Theoretical roofline model of ZCU102 PL based on DSP blocks. The roofline model describes computational ceilings (red and green) and I/O bandwidth ceilings (blue). As the given FPGA does not support floating-point arithmetic, users should consider the number of DSP blocks per operation (green).

100%), so they can change depending on the reasonable amount of resource usage and the maximum achieved frequency. Thus, we compute more realistic maximum computational ceilings determined by the 80% of DSP block utilization following the Register Transfer Level (RTL) synthesis guideline of Vitis HLS and 250 MHz frequency based on the synthesis reports on the 2-D FFT design provided by Vitis HLS. Moreover, we need to consider the number of required DSP blocks per operation, depending on the data type and the data bit-width. If input fixed-point data require only one DSP block for an operation, computational ceiling C_{fix} is:

$$\begin{aligned} C_{fix} &= 2016 \text{ DSP blocks} \times 250 \text{ MHz} \times 1 \text{ op/DSP block} \\ &= 504 \text{ Gops/sec} \end{aligned} \tag{6}$$

As said before, a single-precision floating-point arithmetic operation requires more than one DSP block, especially the floating-point multiplication used in this paper requires 3 DSP blocks per floating-point operation (Flop) based on a Vitis HLS report. The computational ceiling of single-precision floating-point operation C_{fp} is:

$$\begin{aligned} C_{fp} &= 2016 \text{ DSP blocks} \times 250 \text{ MHz} \times \frac{1 \text{ Flop}}{3 \text{ DSP blocks}} \\ &= 168 \text{ GFlops/sec} \end{aligned} \tag{7}$$

In addition, the three FPGA memory interfaces are taken into account for the I/O bandwidth ceiling: i) PS DDR4 to PS, ii) AXI interface between PS and PL, and iii) PL DDR4 to PL. In this study, we consider hardware accelerators whose input data are initially stored in the PS DDR4 and controlled by the PS. Therefore, the I/O bandwidth ceiling is the smaller value between the PS DDR4 bandwidth and PS-PL AXI interface [27]. The PS DDR4 bandwidth (BW_{PS_DDR4}) is the multiplication of DDR transfer rate and DDR width [26]. The Zynq Ultrascale+ has 4 AXI ports [28], and the AXI bandwidth (BW_{AXI}) depends on the clock frequency, transfer data bit-width, and the number of used ports. i.e., the I/O bandwidth ceiling BW is calculated as follows:

$$\begin{aligned} BW_{PS\_DDR4} &= 2133 \text{ MT/sec} \times 64 \text{ bits/transfer} \\ &= 17.064 \text{ GB/sec} \end{aligned} \tag{8}$$

$$\begin{aligned} BW_{AXI} &= 250 \text{ MHz} \times 128 \text{ bits} \times 4 \\ &= 16 \text{ GB/sec} \end{aligned} \tag{9}$$

$$\begin{aligned} BW &= min(BW_{PS\_DDR4}, BW_{AXI}) \\ &= 16 \text{ GB/sec} \end{aligned} \tag{10}$$

D. Parallelism and pipelining

We designed the 2-D FFT to compute both the row-wise FFT and the column-wise FFT by reading all the input data stored in PS DDR4 in the beginning and using only on-chip memory during the FFT, as the FFT size per data set (256 $\times$ 256 and 64 $\times$ 64) fits in the BRAM of the platform[1]. As a reference, we used the Super Sample data Rate (SSR) 2-D FFT provided by the Xilinx open-source HLS-based DSP library [29] by optimizing HLS pragmas and configuring the radix-4 FFT. The original reference features high parallelism and pipelining appropriate to low-latency applications. For example, the reference includes full loop unrolling using `#pragma HLS unroll` with `#pragma HLS array_partition` and task pipelining through `#pragma HLS dataflow` to maximize throughput. However, the library should be modified for the ZCU102 board as the reference design targets Xilinx data center boards, which have larger resources and a different type of on-chip memory. For instance, as Zynq UltraScale+ ZU9EG does not include Ultra RAM (URAM), the `#pragma HLS resource` with the URAM option is removed. Moreover, the floating-point design results in over 100% of DSP block utilization with the FFT size 256 $\times$ 256, so we modified pragmas such as `#pragma HLS pipeline`.

E. FPGA roofline model for the 2-D FFT

We performed RTL synthesis after C synthesis, which shows more realistic results by including placement and routing. Table IV shows the RTL synthesis results on the optimized cases. The floating-point designs use more than 4 times of DSP blocks because a single-precision floating-point multiplication requires 3 DSP blocks per operation. The BRAM utilization is higher with fixed-point data than with floating-point data because we increased the II using `#pragma HLS pipeline` for the floating-point application. The modification makes the number of DSP blocks fit in the given platform. Thus, we recommend using the maximum fixed-point data available in a DSP block for resource efficiency in the case of an actual space-qualified MPSoC, which has fewer resources than industrial-grade products.

[1]At this stage of the work, we focus on the computing performance inside the accelerator. So we consider all data fit in on-chip memory, i.e., I/O bandwidth to access data stored in off-chip memory does not limit the performance

TABLE IV
RESOURCE UTILIZATION OF THE 2-D FFT RTL SYNTHESIS RESULTS INCLUDING PLACEMENT AND ROUTING

FFT size	64 x 64		256x256	
Data type	27-bit FxP	FP32	27-bit FxP	FP32
DSP blocks	196 (7.8%)	928 (37%)	308 (12%)	1276 (51%)
BRAM	48 (2.6%)	32 (1.8%)	618 (34%)	528 (29%)
LUT	58843 (21%)	109870 (40%)	89508 (33%)	157710 (58%)
FF	82129 (15%)	199467 (36%)	103394 (19%)	262969 (51%)
Frequency	225 MHz	227 MHz	224 MHz	217 MHz

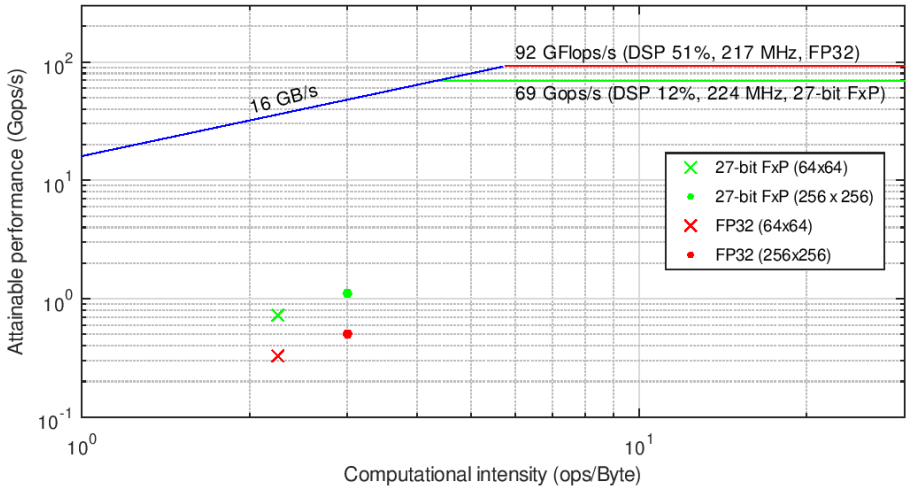

Fig. 5. Roofline model of ZCU102 PL based on DSP blocks. The roofline model consists of the I/O bandwidth ceiling (blue) and the computational ceilings (red and green) based on the RTL synthesis results of 2-D FFT with the FFT size 256 × 256. The designs present the results of different FFT sizes and data types.

Fig. 5 illustrates the 2-D FFT roofline model and the accelerator designs, with different data types and FFT sizes, based on the RTL synthesis results of the input data sets. In this case, the computational intensity of kernel is calculated with the number of multiplications of the radix-4 2-D FFT algorithm and the input data accessed through the AXI interface. In addition, the performance of each design is the ratio between the number of multiplications and the total execution latency. For example, for the fixed-point design with the FFT size 256 × 256, the total latency is 1.5 ms. In addition, the total latency for single-precision floating-point data is 3 ms. The I/O bandwidth ceiling is achieved using the 4 AXI ports. The computational ceilings are obtained from the DSP block utilization, the implemented frequency, and the number of required DSP blocks per multiplication with the given data type. As depicted in the figure, all the designs are under the I/O bandwidth ceiling.

Although the accelerator architectures are highly pipelined and satisfy the algorithm requirements in terms of time, data accuracy, and image size, the performance of the designs is far from the ceiling. This signifies that the platform is oversized or that the FFT code does not efficiently use the available resources, e.g., non-operational behaviors in the FFT code consume significant latency. The used 2-D FFT designs perform matrix transpose between the row-wise FFT and the column-wise FFT to avoid column-wise memory accesses which can be a computational bottleneck. However, the transpose requires memory transfers three times [14], which stalls operations and increases the total latency. Therefore, there is some scope to improve the efficiency of the accelerator architecture by optimizing the memory accesses of the 2-D FFT.

V. CONCLUSION

We present the payload data processing hardware used in the last decade and near-future missions to analyze the on-board processing trends. Considering the performance of state-of-the-art space-qualified MPSoCs, it is expected to migrate payload data processing pipelines partially to on-board systems. The survey on key algorithms reflects the needs of payload teams, so it can be helpful for the space engineering community to prepare high-performance on-board processing for future missions. We

propose combining the roofline model with HLS-based DSE to conduct performance analysis. We finally suggest the roofline of 2-D FFT on the Xilinx Zynq UltraScale+ ZCU102 evaluation board with HLS synthesis results as a case study. The results show the 2-D FFT takes less than 5 ms of latency satisfying the given input data requirements, which are 20 times faster than the assumed time requirement. This significant processing time improvement can be a major factor to consider on-board data processing in future missions.

ACKNOWLEDGMENT

The authors thank Mickaël Bruno and Clément Coggiola at CNES for assistance in prioritizing space scientific algorithms. We also thank several research institutes including CNES, LESIA (Laboratory of Space Studies and Instrumentation in Astrophysics) of the Paris Observatory/CNRS, IJCLab (Laboratoire de Physique des 2 infinis Irène Joliot-Curie) of IN2P3 (National Institute of Nuclear and Particle Physics), and LPC2E (Laboratoire de Physique et de Chimie de l'Environnement et de l'Espace) for participation in our survey to collect payload data processing algorithms.

REFERENCES

[1] N. Montealegre, D. Merodio, A. Fernández, and P. Armbruster, "In-flight reconfigurable FPGA-based space systems," in *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Jun. 2015, pp. 1–8.

[2] P. Plasson, G. Brusq, F. Singhoff, H. N. Tran, S. Rubini, and P. Dissaux, "PLATO N-DPU on-board software: an ideal candidate for multicore scheduling analysis," in *11th European Congress ERTSS Embedded Real Time Software and System*, Toulouse, France, 2022.

[3] F. Torelli, "Common DPU and Basic Software for JUICE Instruments," in *European Workshop on On-Board Data Processing (OBDP2019)*, European Space Research and Technology Centre (ESTEC), Feb. 2019.

[4] M. Maksimovic *et al.*, "The Solar Orbiter Radio and Plasma Waves (RPW) instrument," *Astronomy & Astrophysics*, vol. 642, p. A12, Oct. 2020. [Online]. Available: https://www.aanda.org/10.1051/0004-6361/201936214

[5] W. Benz *et al.*, "The CHEOPS mission," *Experimental Astronomy*, vol. 51, no. 1, pp. 109–151, Feb. 2021. [Online]. Available: https://doi.org/10.1007/s10686-020-09679-4

[6] Y. Kasaba *et al.*, "Mission Data Processor Aboard the BepiColombo Mio Spacecraft: Design and Scientific Operation Concept," *Space Science Reviews*, vol. 216, no. 3, p. 34, Apr. 2020. [Online]. Available: http://link.springer.com/10.1007/s11214-020-00658-x

[7] J. H. J. d. Bruijne, M. Allen, S. Azaz, A. Krone-Martins, T. Prod'homme, and D. Hestroffer, "Detecting stars, galaxies, and asteroids with Gaia," *Astronomy & Astrophysics*, vol. 576, p. A74, Apr. 2015. [Online]. Available: https://www.aanda.org/articles/aa/abs/2015/04/aa24018-14/aa24018-14.html

[8] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009. [Online]. Available: https://dl.acm.org/doi/10.1145/1498765.1498785

[9] M. Siracusa *et al.*, "A comprehensive methodology to optimize fpga designs via the roofline model," *IEEE Transactions on Computers*, vol. 71, no. 8, pp. 1903–1915, 2022.

[10] N. Ibellaatti *et al.*, "HERMES: qualification of High pErformance pRogrammable Microprocessor and dEvelopment of Software ecosystem," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Apr. 2023, pp. 1–5.

[11] N. Perryman, C. Wilson, and A. George, "Evaluation of Xilinx Versal Architecture for Next-Gen Edge Computing in Space," in *2023 IEEE Aerospace Conference*, Mar. 2023, pp. 1–11.

[12] S. Wiehle, S. Mandapati, D. Günzel, H. Breit, and U. Balss, "Synthetic Aperture Radar Image Formation and Processing on an MPSoC," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[13] J. S. Smith, B. H. Dean, and S. Haghani, "Distributed computing architecture for image-based wavefront sensing and 2D FFTs," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, H. Lewis and A. Bridger, Eds., vol. 6274, Jun. 2006, p. 627421.

[14] C.-L. Yu, J.-S. Kim, L. Deng, S. Kestur, V. Narayanan, and C. Chakrabarti, "FPGA Architecture for 2D Discrete Fourier Transform Based on 2D Decomposition for Large-sized Data," *Journal of Signal Processing Systems*, vol. 64, no. 1, pp. 109–122, Jul. 2011. [Online]. Available: http://link.springer.com/10.1007/s11265-010-0500-y

[15] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965. [Online]. Available: https://www.ams.org/mcom/1965-19-090/S0025-5718-1965-0178586-1/

[16] R. Neuenfeld, M. Fonseca, and E. Costa, "Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time," in *2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS)*, Feb. 2016, pp. 171–174.

[17] B. C. Schafer and Z. Wang, "High-Level Synthesis Design Space Exploration: Past, Present, and Future," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2628–2639, Oct. 2020.

[18] B. da Silva Gomes, A. Braeken, E. D'Hollander, and A. Touhafi, "Performance modeling for FPGAs: extending the roofline model with high-level synthesis tools," *International Journal of Reconfigurable Computing*, vol. 2013, pp. 1–10, 2013. [Online]. Available: http://hdl.handle.net/1854/LU-4226966

[19] J. Cong *et al.*, "FPGA HLS Today: Successes, Challenges, and Opportunities," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 4, pp. 1–42, Dec. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3530775

[20] "Vitis High-Level Synthesis User Guide (UG1399)," AMD, Tech. Rep., 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1399-vitis-hls/Introduction

[21] R. Kastner, J. Matai, and S. Neuendorffer, "Parallel Programming for FPGAs," May 2018. [Online]. Available: http://arxiv.org/abs/1805.03648

[22] M. Fingeroff, *High-level Synthesis: Blue Book*. Xlibris Corporation, 2010.

[23] S. Schanne *et al.*, "A Scientific Trigger Unit for space-based real-time gamma ray burst detection I - Scientific software model and simulations," in *2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (2013 NSS/MIC)*, Oct. 2013, pp. 1–5.

[24] N. Dagoneau, "Détection de sursauts gamma ultra-longs et traitement d'images embarqué pour le télescope spatial SVOM/ECLAIRs," Theses, Université Paris-Saclay, Oct. 2020. [Online]. Available: https://theses.hal.science/tel-03009638

[25] D. Diakite, N. Gac, and M. Martelli, "OpenCL FPGA Optimization guided by memory accesses and roofline model analysis applied to tomography acceleration," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, Aug. 2021, pp. 109–114.

[26] "Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics (DS925)," AMD, Tech. Rep., 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/ds925-zynq-ultrascale-plus

[27] E. Calore and S. F. Schifano, "Performance assessment of FPGAs as HPC accelerators using the FPGA Empirical Roofline," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. Dresden, Germany: IEEE, Aug. 2021, pp. 83–90. [Online]. Available: https://ieeexplore.ieee.org/document/9556357/

[28] K. Manev, A. Vaishnav, and D. Koch, "Unexpected Diversity: Quantitative Memory Analysis for Zynq UltraScale+ Systems," in *2019 International Conference on Field-Programmable Technology (ICFPT)*. Tianjin, China: IEEE, Dec. 2019, pp. 179–187. [Online]. Available: https://ieeexplore.ieee.org/document/8977835/

[29] (2023) Vitis Accelerated Libraries. [Online]. Available: https://github.com/Xilinx/Vitis_Libraries