BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 69 (73), Numărul 2, 2023
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ
DOI:10.2478/bipie-2023-0012

DOMAIN-SPECIFIC FPGA-BASED DIGITAL ACQUISITION SYSTEM FOR HARDWARE ACCELERATORS INTEGRATION

BY

COSMIN-ANDREI POPOVICI*, ANDREI STAN and VASILE-ION MANTA

“Gheorghe Asachi” Technical University of Iași,
Faculty of Automatic Control and Computer Engineering, Iași, Romania

Received: June 20, 2024
Accepted for publication: July 25, 2024

Abstract. Domains like IoT (Internet of Things), ADAS (Advanced Driver Assistance System), Smart Homes and Smart Cities and XaaS (Everything-as-a-Service) put a great pressure in the last fifteen years on computer engineers for designing processors capable of higher precision computations performed in shorter times, interconnected in ultra-high-speed communication networks and consuming less electrical power in order to be supplied by batteries and clean renewable energy solutions.

For almost two decades now, classical VLSI laws such as Moore’s and Dennard’s cannot be applied anymore for embedding twice as many transistors in the same silicon area every two years and keeping the energy density constant at the same time. The most feasible solution for mitigating this problem is designing DSAs (Domain-Specific Architectures) like application-specific hardware accelerators. Hardware accelerators replace the necessity of implementing domain specific algorithms programmatically by embedding hardware implementation variants of the same algorithms in modules running in parallel with the software developed for classical CPUs.

*Corresponding author; *e-mail*: cosmin-andrei.popovici@academic.tuiasi.ro
© 2023 Cosmin-Andrei Popovici et al.
This is an open access article licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

An electronic field which can benefit from using domain-specific architectures instead of classical general-purpose microcontrollers is Digital Signal Processing (DSP). Designing logical circuits or complex digital systems won't be possible today without using logic analyzers, protocol decoders, digital oscilloscopes, and digital recorders. These tools are used in the processes of prototyping and verifying digital systems by recording, displaying, decoding, and analyzing internal or external signals.

This article proposes an example of a domain-specific architecture, a Digital Acquisition System, named FpgaDaqEth, acquiring data from 32 digital channels at 1 MS/s maximum acquisition rate, deployed on FPGA, operating at 100 MHz, using DDR3 RAM memory and 100 Mbps Ethernet for the communication with hosting PC running a custom GUI (graphical user interface) application. This design offers between 8x and 43.34x better propagation delay (constant 20 ns) for performing and outputting logical operations over multiple inputs (between 2 and 31 signals) than a general-purpose microcontroller which runs at 6x times higher frequency. It is also initiating UDP transmissions containing digital records and their timestamps 19.31x times faster than a Xilinx (now AMD) MicroBlaze processor implementation running at 2x times higher frequency on the same FPGA. The solution can detect changes in parameters of PWM inputs and send them to the hosting PC right after the first cycle of the modified signal ends.

The proposed design is deployed on AMD XC7A100T FPGA as a standalone solution and may be integrated as processor extension for RISC-V cores in a future project.

Keywords: VLSI, DSA, FPGA, digital signal acquisition, processors.

1. Introduction

The technologies and domains which promised and already managed to improve and ease people's life in the last decade, like IoT (Internet of Things), ADAS (Advanced Driver Assistance System), Smart Homes and Smart Cities and XaaS (Everything-as-a-Service) rely not only on extremely optimised software, but on complex hardware systems. These complex systems include analog and digital subsystems for providing high-speed communication interfaces, high-performance and high-precision computing, distributed computation over multiple computing units, real-time decision-making processes, real-time failure detection and low-power and battery management features.

Taking into consideration the ECUs (electronic control units) present in modern car and developed into the automotive industry, it can be observed that one of these systems has tens of digital and analog inputs, tens of communication interfaces (CAN, CAN-FD, LIN, FlexRay, Ethernet, etc.) and between 100 and 200 output lines.

IoT (Internet of Things) and Smart Systems also rely on a big number of digital inputs representing sensors data and messages between multiple nodes

vehiculated over wired or wireless transmission environments and on a big number of outputs comprising actuators and communication peripherals.

Debugging of digital circuits and complex digital systems requires tools such as oscilloscopes, logic analysers and protocol decoders. Popular measurement tools like these offer parallel acquisition on 2, 4, 6 or 8 analog channels and 8 or 16 digital channels, in case of oscilloscopes and 8, 16, 24 or 32 digital channels in case of logical analysers and protocol decoders. This type of solutions varies in price from hundreds to tens of thousands of euros and don't always fit the digital designers needs in term of number of channels, signal acquisition frequencies, communication interfaces or test toolchain integrability. In this situation, digital designers and embedded applications developers prefers to build their own custom digital acquisition solutions using microcontrollers or FPGAs (Field-Programmable Gate Arrays).

At the moment of writing this paper, the digital designers and developers are forced to make a paradigm shift from general-purpose solutions to domain-specific architectures in order to mitigate the problems created by already reaching the technological limitations of silicium – it's already difficult and it will be impossible in the future to shrink more transistors in the same circuit area for gaining higher operating frequencies. These deviations from the historical VLSI (very large-scale integration) principles are pushing researchers to make a transition from general-purpose designs to domain-specific architectures in order to increase the performance and reduce the power consumption of digital systems.

Digital acquisition and control represent a domain which certainly benefit from domain-specific architectures. In this context, the current paper proposes a digital acquisition and digital control system, based on FPGA and using 100 Mbps Ethernet as communication medium, named FpdaDaqEth.

The proposed solution is able to individually configure up to 32 channels as digital inputs or digital outputs, to filter or negate individual channels, to select the digital output source between values given directly by user, logical operations between inputs or PWM generated signals and to perform digital acquisition to attached DDR3 memory or directly to hosting PC's RAM via Ethernet connection.

Even though the proposed solution operates at a main clock of 100 MHz, it manages to offer a better and constant response time for performing logical operations for between 2 and 31 inputs between 8x and 43.3x times faster than a Cortex-M7 microcontroller which operates at a 6x times higher frequency. It also sends Ethernet frames containing signals changes 19.31x faster than an AMD MicroBlaze CPU operating at 2x higher frequency and deployed on the same FPGA, AMD Artix-7. PWM measurement is performed faster than using 2x – 5x times faster than in the case of using the Cortex-M7 running at 600 MHz and the changes in the parameters of the acquired PWM signals is detected after just one cycle of the updated signals.

The proposed solution is presented in terms of architecture and functioning principles alongside the PC application developed for using the digital acquisition and control system. The PC application has two components. The first is a DLL (dynamic-link library) containing the Ethernet/UDP based driver for controlling the proposed solution (getting all channels values, setting filters, negating inputs, setting outputs, starting/stopping recording) and for debugging it. The second component is the GUI (graphical user interface) used for performing the actions described above in the driver's description and visualizing the signal plots or the table/chart generated by driver based on the signals data acquired.

The first section of the article represents the introduction, the second one briefly describes some theoretical information on signals and signal acquisition, the third one summarizes some related work, the state of the art in digital acquisition and some available market solutions used for digital acquisition (microcontrollers which can be used for digital acquisition and/or digital control). The fourth part illustrates the architecture of the proposed solution and the fifth one focuses on how the solution was tested and validated and which criteria were used for this purpose. The sixth section shows how the digital acquisition system developed as a domain-specific system will be transformed into a RISC-V extension into a future project and also proposes an extension ISA. The seventh part shows the software developed for using and debugging the solution. The eighth section summarizes the conclusions.

2. Few words on Signals and Signal Acquisition

There are multiple ways of defining signals. From a mathematical point of view a signal is defined as "physical quantity that varies with time, space or other independent variable or variables" (Proakis and Manolakis, 1995, p. 2). From the point of view of their fields of application, signals are defined as "any measurable quantity that conveys information […] (e.g., the voltage output of an amplifier); acoustical signals (the pressure measured by a microphone); mechanical signals (the force measured by a strain gauge); biological signals (body temperature, blood pressure, waveforms of EKG, EEG); financial (bank balance, stock price)" (Holton, 2021, p. 6).

The signals acquired by the digital acquisition systems like the system proposed in the current paper are electrical signals – voltages – which are periodic or non-periodic functions (or combination of such functions) in the time domain – as illustrated in Eq. (1):

$$V = f(t) \tag{1}$$

where V is the amplitude of the voltage, $f(t)$ is the function describing the evolution of the voltage and t is the time, the only independent variable influencing the evolution of the function.

As stated also by Holton (2021, p. 6), there are four categories of signals categorized by their function's domains and codomains:

- Continuous-time, continuous-amplitude signals: both time and amplitude belong to domain with infinite number of elements – time is *unquantized*, so the signal can be measured at any moment in time; this is the *analog signal* (example in Fig. 1a).
- Discrete-time, continuous-amplitude: time takes values only from a finite number of values, so the signal is defined at specific moments – time is quantized – and the amplitude values belong to an infinite set of values; this is called a *digital signal* (example in Fig. 1b).
- Continuous-time, discrete-amplitude: the signal can be measured at any time instance, but its amplitude values belong to a finite number of values (example in Fig. 1c).
- Discrete-time, discrete-amplitude: both time and amplitude belong to discrete sets of elements (example in Fig. 1d).

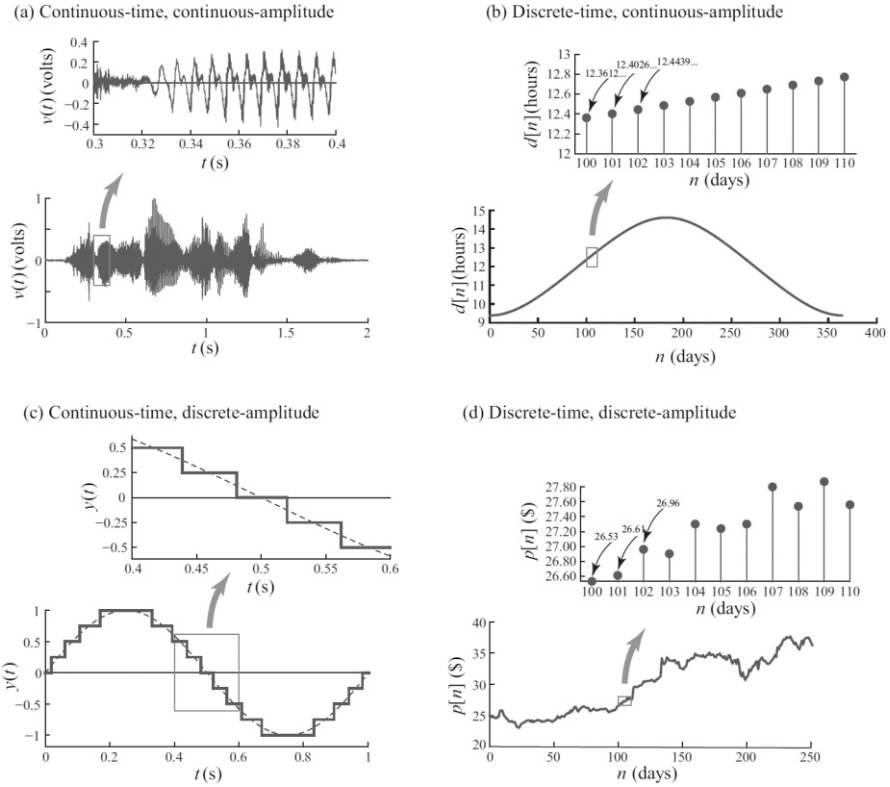


Fig. 1 – Continuous-time and discrete-time signals (Holton, 2021, p. 6).

Due to the fact that the digital signal processors have limited RAM memory and limited capacity for number representation, the signal acquired throw the digital signal acquisition is a *discrete-time, discrete-amplitude signal*. The number of samples which can be stored is dictated by the RAM memory capacity and the number of values which can be taken by the amplitude depends on the resolution of the component responsible for sampling the analog signal, named the Digital/Analog Converter. Both sampling process and D/A converter are described below.

From the point of view of the electronic component performing the acquisition of an analog signal, there are three stages of the procedure: sampling, quantization and coding the sampled signal.

Sampling an analog signal is the process of collecting the values of a continuous signal at integer instants of time, which are multiples of the sampling period T, the rest of values not being used. Sampling of an analog signal $x(t)$, "can be modelled theoretically as the multiplication of the continuous-time signal $x(t)$ by an impulse train $s(t)$, an infinitely long train of impulses that is periodic with period T" (Holton, 2021, p. 334); equation is Eq. (2):

$$x_s(t) = x(t) \times s(t) \tag{2}$$

where $x_s(t)$ is the sampled signal and the $s(t)$ represents the impulse train modelled in Eq. (3):

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{3}$$

where $\delta(x)$ is the Dirac delta function which can be "loosely thought of as a function on the real line, which is zero everywhere except at the origin, where it is infinite…"

$$\delta(x) \cong \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{4}$$

"… and which is also constrained to satisfy the identity" (Gelfand and Shilov, 2014):

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{5}$$

The process of the analog signal sampling as the multiplication of the continuous-time signal and the impulse train is depicted in Fig. 2.

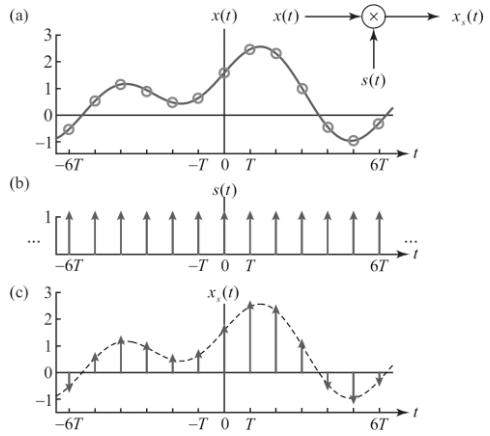


Fig. 2 – Sampling of an analog signal (Holton, 2021, p. 334).

The *quantization* is the process where the sampled signal – discrete-time, continuous-amplitude signal – is transformed into a discrete-time, discrete-amplitude signal. Because an infinite number of amplitude possibilities cannot pe internally represented by the digital signal processor. During the A/D conversion, each amplitude level is approximated by the quantizer to the closest value to be represented on a finite number of bits, called the resolution of the A/D converter (e.g. for a 12-bit A/D converter with the capability to measure voltages between 0 and 3.3V, there will only be $2^{12} = 4096$ possible representable values, so all the values will be placed at equidistant intervals of 3.3V / 4095 ≈ 8.0586 × 10^{-4} V).

The third and final stage of the A/D conversion, the *coding* of the values is performed by the encoder using the resolution of the quantizer to represent the quantized values into binary values used further by the digital signal processor.

The most crucial condition to be taken into consideration for digital signal acquisition is the Sampling Theorem or the Nyquist-Shannon Theorem which states that: "if a function $x(t)$ contains no frequencies higher than B hertz, than it can be completely determined from its ordinates at a sequence of points spaced less than $1/2B$ seconds apart" (Shannon, 1949). In other words, the sample rate (F_S) used for sampling the analog signal should be bigger than the double of the maximum frequency (F_{Max}) present in the original signal, in order to have an accurate representation of the signal in order to reconstruct it, performing measurements or transformations on it, etc., as we can see in Eq. (6):

$$F_S > 2 \times F_{Max} \tag{6}$$

Few paragraphs ago, analog signals were defined as continuous-time, continuous-amplitude signals and digital signals were defined as discrete-time, continuous-amplitude signals obtained by quantizing the sampled signals and stored as values arrays in the memory of digital signal processors.

From the point of view of amplitude's definition domain, analog signals are also the continuous-time, continuous-amplitude signals and the digital signals are continuous-time, discrete-amplitude signals (Fig. 3). The amplitude of the digital signal can take only two values, corresponding to logic "0" and logic "1" symbols. The logical "0" means 0 V, but the physical value of logical symbol "1" depends on the transistors used in the digital circuit: V_{CC} = 5V for TTL and V_{DD} = 1V2, 1V5, 1V8, 2V5, 3V3 or 5V for CMOS families. For CMOS families "0" has a range between 0V and 30% of V_{DD} and "1" has the interval between 70% of V_{DD} and V_{DD}.

The proposed solution is meant to perform digital acquisition and digital control over digital signals considering the definition depicted in the paragraph above, signals having as amplitude just the logical levels "0" and "1".

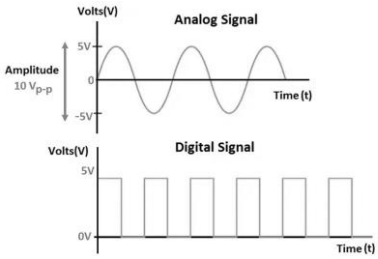


Fig. 3 – Analog Signal vs Digital Signal.

3. Available Market Solutions and State of the Art

Because the proposed solution, the digital acquisition system, is intended to be transformed in processor extension in the future, the available market solutions taken into consideration for comparison with our solution are microcontrollers capable of acquiring digital statuses from a decent number of inputs and capable of transferring them to a PC on communication interfaces which are fast enough for avoiding buffer overrunning or exceeding RAM memory capacity. This means that the transfer data rate should be greater than the data rate generated by sampling all digital inputs (Eq. 7):

$$DR_{communication} > DR_{digital\ acquisition} \tag{7}$$

where $DR_{communication}$ means data transfer rate (bits per second) and $DR_{digital\ acquisition}$ means the data rate of the digital acquisition, which is given by the formula below:

$$DR_{digital\ acquisition} = N_{channels} \times SR \tag{8}$$

where $N_{channels}$ represents the number of digital inputs of the microcontroller and SR represents the sample rate of the digital acquisition.

The first market solution is a 32-bit Cortex-M4 from Infineon, named XMC4500, having 4 ports of 16 digital inputs/outputs and 1 port of 8 digital inputs/outputs. It is also capable of generating up to 4 independent PWM (pulse width modulation) signals. It is relevant for comparison with the proposed solution due to having CAN (1 Mbps) and Ethernet (100 Mbps) communication capabilities.

The second microcontroller used for comparison and benchmarking is NXP MIMXRT1062, a 32-bit Cortex-M7 featuring 5 ports of up to 32 general purpose pins each and various communication interfaces such as CAN (1 Mbps), CAN-FD (8 Mbps) and Ethernet (100 Mbps). This microcontroller can also generate 22 PWM signals routable to 35 pins.

For the third solution, we took an IP (intellectual property) from AMD, a MicroBlaze core, and configured it to work at 200 MHz, to have 32 digital pins in 4 ports of 8 pins each and to offer 100 Mbps Ethernet peripheral. For performing relevant tests and measurements in comparison with the proposed solution, the MicroBlaze core design was uploaded and used on the same FPGA used for our design, a XC7A100T-1CSG324C. The block diagram of the configured IP is presented in Fig. 4.

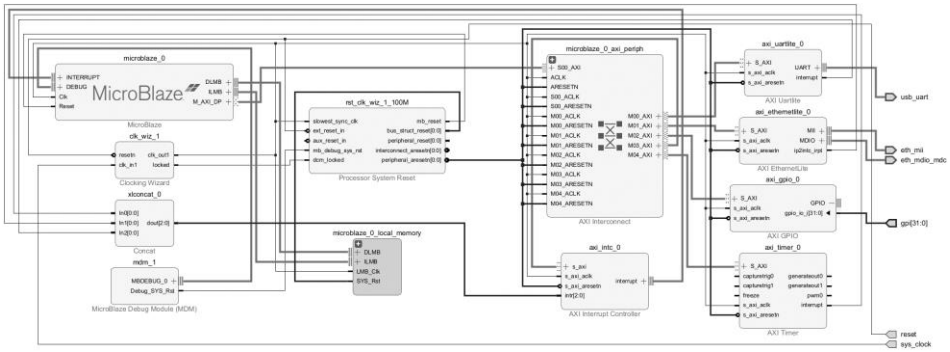


Fig. 4 – Block Diagram of the MicroBlaze design used for comparison with the proposed solution.

Other solutions used for signal acquisition which represent the state of the art and are widely used for designing, debugging, testing and verifying of complex systems are ADALM2000, Saleae Logic Pro and Digilent Discovery 3.

Analog Devices ADALM2000 (AD Active Learning Module) is a digital USB oscilloscope with 2 100 MS/s ADCs, with 2 arbitrary function generator and 2 150 MS/s DACs, 16 digital channels (100 MS/s), 16 channels for pattern generation, two input/output trigger lines for linking multiple devices, 1 channel voltmeter for AC/DC ±25V, network analyser (Bode, Nyquist and Nichols transfer diagrams) for 1 Hz – 10 MHz range and 2 channels programmable power supplies.

Table 1
Comparison between microcontrollers which can be used for digital acquisition and the proposed solution

Features \ Devices	Infineon XMC4500	NXP MIMXRT1062	MicroBlaze core*	**FpgaDaqEth**
Architecture	32-bit Cortex-M4	32-bit Cortex-M7	32-bit RISC	**DSA**, non-CPU**
Frequency	120 MHz	600 MHz	200 MHz	**100 MHz**
Ports of 8 digital IO/s	1	0	4	**0**
Ports of 16 digital IO/s	4	0	0	**0**
Ports of 32 digital IO/s	0	5	0	**1**
PWM generation	4 generators	22 generators	32 generators	**32 generators**
UART	4 channels	8 channels	N/A	**1 for debug**
CAN	2 channels, CAN 2.0	2 CAN2.0 & 1 CAN-FD	N/A	**N/A**
Ethernet	10/100 Mbps	10/100 Mbps	10/100 Mbps	**10/100 Mbps**
*customized by authors for comparison with proposed solution, deploying both on same FPGA				
**domain-specific architecture				

Saleae Logic Pro 8 and Saleae Logic Pro 16 offer digital acquisition at maximum sample rate of 500 MS/s (mega samples per second) for 8 or 16 signals of maximum 100 MHz for 1V2, 1V8 and 3V3 TTL (transistor-transistor logic) families and analog acquisition at a maximum sample rate of 50 MS/s, for signals of maximum 5 MHz, with input voltages between -10 V and 10 V at a resolution of 12 bits and an attenuation of -50 dB or better. Its channels are featuring 2 MΩ input impedance, input capacitance of 10 pF and an input voltage protection of ±25V. One of the most important features is the capacity of protocol decoding supported by software extensions of the PC application developed by the producer or by third parties.

Digilent Analog Discovery 3 is and 125 MS/s USB oscilloscope, waveform generator, logic analyser and variable power supply, digital acquisition of 16 channels at up to 125 MS/s sample rate, differential analog acquisition on 2 channels at 14-bit resolution, ±25V input voltage range for signals at up to 30 MHz bandwidth. This tool is also capable of protocol decoding. The additional features of pattern generation and variable power supplying prove to be extremelly useful in advanced debug or test sessions where the test condition implies stimulus for the DUT (device under test).

Table 2

Comparison between of-the-shelf digital acquisition solutions and the proposed solution

Features \ Devices	AD ADALM2000	Saleae Logic Pro 8/16	Digilent Analog Discovery 3	**FpgaDaqEth (proposed solution)**
Channels	16	8/16	16	**32**
Digital Sample Rate	125 MS/s	500 MS/s	125 MS/s	**100 MS/s**
Analog Sample Rate	100 MS/s	50 MS/s	125MS/s for 2 diff pairs	**N/A**
Connectivity	USB 2.0	USB 3.0	USB 3.0	**Ethernet 100 Mbps**
Pattern Generation	150 MS/s	No	2 channels	**32 channels, only PWMs**
Variable Power Supply	2 channels	No	2 channels	**No**
Triggers	1 input and 1 output	No	No	**No**

4. Architecture of the Proposed Solution

As discussed in the introduction, the proposed solution is a Digital Acquisition System, named FpgaDaqEth, capable of both reading and writing the states of 32 individual Digital IOs. It is also capable of recording the digital IOs selected by the user and any IO configured as input can be individually negated. Recording is performed into the attached memory of the FPGA deploying the design or into the RAM memory of the PC connected to the solution. For setting the values of the IOs configured as outputs, there are three options: using the values given by the user from an output mask, mirroring the result of logical operations performed on certain inputs of the system or outputting a PWM signal configured by the user.

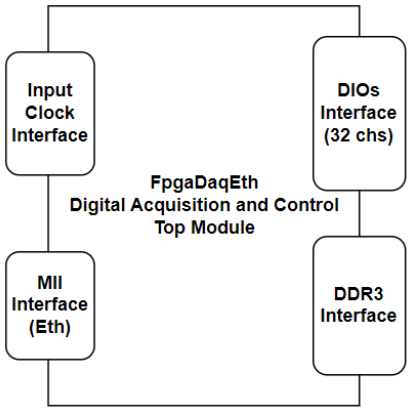


Fig. 5 – FpgaDaqEth – Top Module and its main interfaces.

The main interfaces of the FpgaDaqEth are illustrated in Fig. 5. The first interface is the Input Clock Interface. This interface receives the clock signal from an onboard MEMS (microelectromechanical system oscillator) generating a 100 MHz clock. This clock will be internally divided by the Clocking Wizard module into various frequencies needed by the submodules of the design.

The MII Interface contains all the signals connecting the Ethernet/IPv4/UDP stack IP and the onboard Ethernet PHY (100 Mbps). This interface is responsible for the communication of the solution with the hosting PC running the FpgaDaqEth application.

The DDR3 Interface represents the communication bridge between the FPGA deploying the solution and attached onboard 256 MB DDR3 RAM memory used for recording the Digital IOs changes and their timestamps.

The last interface is the Digital IOs interfaces composed by 32 input/output channels configured by the FPGA constraints with the LVCMOS33 standard (3V3 CMOS). These are the channels written, read and sampled for digital signal acquisition by the solution.

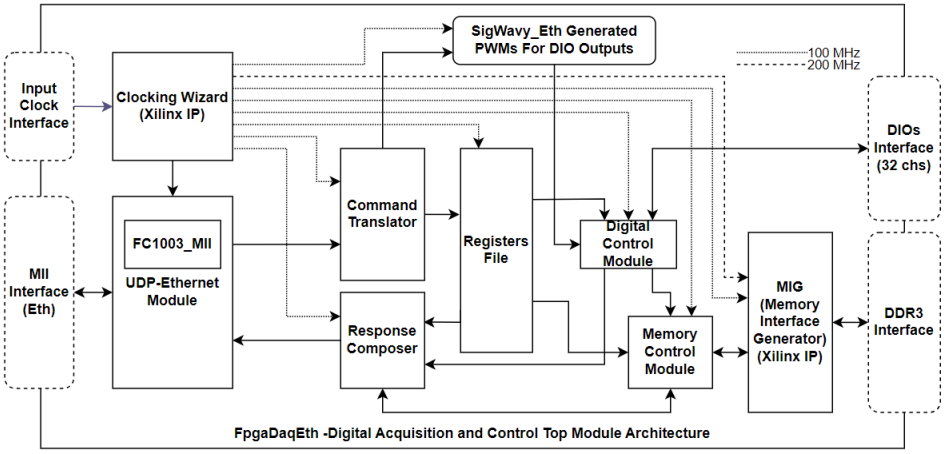


Fig. 6 – Architecture of FpgaDaqEth – Main Modules.

The main modules of the FpgaDaqEth digital acquisition system are illustrated in Fig. 6 and they have the purposes described below.

ClockingWizard is an AMD IP which takes one input from the Input Clock Interface, the 100 MHz clock from the MEMS oscillator and it's configured to generate two clocks: *clk_out_100* of 100 MHz used by all the modules in the design and *clk_out_200* of 200 MHz used as reference clock by the MIG (Memory Interface Generator), module used for controlling the DDR3 RAM operations.

The UDP-Ethernet Module is a wrapper module around FC1003_MII, also used by authors in paper (Popovici and Stan, 2023). FC1003_MII is an IP distributed as a black-box netlist which implements the whole OSI communication protocols including Ethernet (Hornig, 1984), IPv4 (Postel, 1981a), UDP (Postel, 1980), ARP (Plummer, 1982) and ICMP (Postel, 1981b) and also an UDP server accessible through configurable IPv4 address and port number or DHCP (Droms, 1997) which exposes a bytes send/receive interface with *valid, last and ready* signals and 8-bit data buses for payload data.

CommandTranslator module receives UDP payloads, consisting in 10 bytes (Function Code byte and 9 command specific data bytes) and transforms them into signals and buses used for controlling write/read operations over RegistersFile module and for controlling the enabled/disabled states and parameters of the PWMs generated by PWM generation unit described in paper (Popovici *et al*., 2023).

One of the modules receiving commands from CommandTranslator is RegistersFile module. RegistersFile module acts as a Control and Status Register containing 64 registers of 32-bit word length (Table 3). Specific registers can be written/read for storing/retrieving the configurations of:

- Digital Control Module: the mask for direction of each channel, inputs filter and inputs negations mask, output source mask, masks for forcing outputs to take values from PWMs generated signals or from logical operations performed on certain inputs, masks for configuring which channels are used in AND, OR or XOR operations, bits for enabling/disabling recording to attached RAM or to PC's RAM, etc.
- Memory Control Module: enable/disable DDR3 memory operations, issuing DDR3 memory clear/reset, maximum memory.

The only registers which are read-only are the identification and versions (HW and Bitfile, major + minor) registers and the registers storing the PWM measurements data of all 32 channels.

Table 3

Addresses, Names and Purposes of the Registers in Registers File Module

Address	Name	Fields
0	Identification	32-bit word that never changes
1	Versions	16 bits for HW version and 16 bits for Bitfile version
2	Digital Acquisition Parameters	2 bits for enabling/disabling recording to attached RAM or to PC's RAM, 2 bits for selecting sample rate (25 MHz, 50 MHz or 100 MHz)
3	Direction Mask	32-bit mask for individually configuring channels directions
4	Output Mask	32-bits mask for output values
5	Filter Mask	32-bit mask selecting which channels are used for recording
6	Negate Mask	32-bit mask selecting which inputs to be negated

7	AND Selection Mask	32-bit mask for selecting which channel should be included into a multi-channel AND logical operation
8	OR Selection Mask	32-bit mask for selecting which channel should be included into a multi-channel OR logical operation
9	XOR Selection Mask	32-bit mask for selecting which channel should be included into a multi-channel XOR logical operation
10	Overwritten Output Channels for Logical Operations Results	3 8-bits addresses of the output channels overwritten with the result of multi-channel AND, OR and XOR operations
11	Memory Operations and Parameters	1 bit for enabling/disabling DDR3 RAM operations, 1 bit for cleaning/resetting the contents of DDR3 and 24 for bits for the maximum number of 128-bit memory words to be written
12	reserved	reserved
…	…	…
31	reserved	Reserved
32	PWM Data from Channel 0	16 bits for High Ticks and 16 bits for Low Ticks registered on Channel 0
…	…	…
63	PWM Data from Channel 31	16 bits for High Ticks and 16 bits for Low Ticks registered on Channel 31

The module responsible for digital acquisition and digital control is Digital Control Module (Fig. 7). It interacts with two interfaces of top module, Digital IOs Interface and the Memory Control Interface and with three internal interfaces: RegistersFile Configurations Interface for receiving it configurations, Generated PWMs Interface for routing certain generated PWMs on certain outputs, RegistersFile Measured PWMs Interface for writing back to RegistersFile all the measured input PWMs and Response Composer Interface, used for sending digital acquired signals states to PC via UDP/Ethernet.

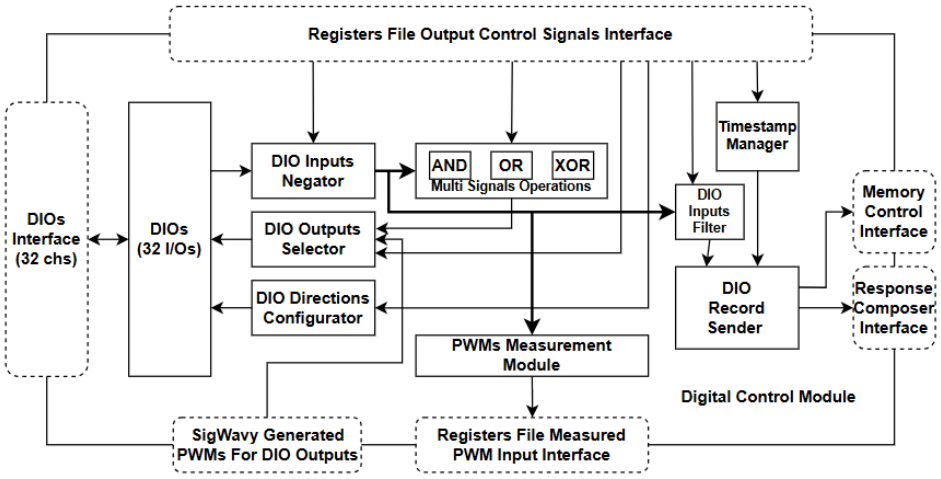


Fig. 7 – Digital Control Module - Architecture Overview.

The Digital Control Module contains the following modules:

- DIOs Block – the 32 in/out general-purpose digital channels configurable as inputs or outputs.
- DIO Directions Configurator – submodule used for individually configuring the directions of the digital channels.
- DIO Inputs Negator – submodule used for negating certain inputs selected by user for the user to easily analyse the recorded waveforms.
- DIO Outputs Selector – used for routing given values states, results of multi-bit logical operations or generated PWM signals to selected output channels.
- Multi Signal Operations – submodule used for performing AND, OR or XOR on the inputs selected by user through multi-channel operations masks registers.
- PWM Measurement Module – this submodule contains PWM measurements blocks for all 32 channels and writes back the numbers High Ticks (16 bits) and Low Ticks (16 bits) of every channel to their specific register in RegistersFile.
- DIOs Input Filter submodule is used for filtering certain inputs before they reach the recording stage in order to not consider the changes occurred in non-relevant inputs.
- Timestamp Manager is a 32-bit timer giving the microseconds timestamp of the recorded signal changes.
- DIO Record Sender submodule samples all the channels and, whenever a change appears in the filtered channels, it sends the current state together with the 32-bit timestamp (microseconds) to the Memory Control Module (for writing them to the attached DDR3 RAM) or to the Response Composer Interface (for encapsulating them into UDP frames for sending them to PC).

The module handling all the communication with the DDR3 RAM memory attached to the FPGA deploying the design of the proposed solution is the Memory Control Module (Fig. 8). It takes its two clocks from the Clocking Wizard Interface, the 100 MHz clock generally used by all modules of the solution for FSMs (finite state machines) and the 200 MHz clock used as reference clock by the AMD MIG IP (Memory Interface Generator).

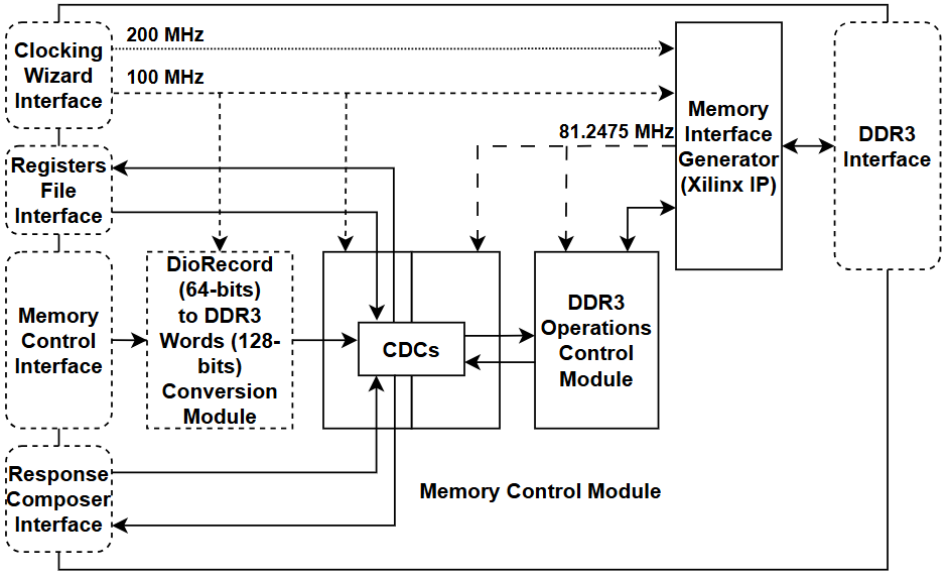


Fig. 8 – The Architecture of the Memory Control Module.

The commands for starting the writing of the signal states together with their timestamps to DDR3 memory, for reading the values back to user logic for sending all acquired data to PC and for cleaning/resetting the memory are taken from the Registers File Interface. The Memory Control Interface feeds the module with the recorded samples to be written to memory in words of 128 bits each. The Response Composer Interface receives word by word all the data from the DDR3 memory after an acquisition cycle is finished in order to transfer all data to the hosting PC. The DDR3 interface is formed by all the classic RAM access control signal and data buses, and it's handled by the MIG module.

The Memory Control Module is separated into two clock domains. The first one is the general clock domain of the solution, 100 MHz and the second one uses the frequency exposed by the MIG (Memory Interface Generator). MIG derives the clock used for RAM operations from the 100 MHz clock, the resulting DDR3 clock being 649.98 MHz, which means that rising edge operations and falling edge operations has a clock of 324.99 MHz each (Eq. (9)). The clock exposed by MIG for user operations is a quarter of edge operation clock, so 81.2475 MHz, as it is calculated in Eq. (10).

$$F_{edge} = \frac{F_{DDR}}{2} = \frac{649.98\ MHz}{2} = 324.99\ MHz \tag{9}$$

$$F_{user\ operations} = \frac{F_{edge}}{4} = \frac{324.99\ MHz}{4} = 81.2475\ MHz \tag{10}$$

Since the general clock of the proposed solution is 100 MHz and the clock exposed by MIG for taking write/read commands which will be translated to RAM operations is 81.2475 MHz, Clock Domain Crossing (CDC) modules together with controlling FSMs were required. The CDCs are bidirectional and expose data buses and control signals which are used by the designers of the solution for exchanging data between the two different clock domains mentioned above.

The MIG was configured for interacting with a 256 MB Micron MT41K128M16, working at 1.35V, exposing 27 bits bus for address (3 for the bank, 14 for the row and 10 for the colon), 16 bits as data bus (8 16-bit words, so 128-bit word at every operation), 50 Ohms internal termination impedance for the high range banks and sequential burst of strict ordered operations.

The module used for generating PWM signals for routing them as outputs is a modified version of the PWM generation unit described by the authors in paper (Popovici et al, 2023) and it is configured by writing into the memory of the desired channel two 32-bit words representing the numbers of high ticks and low ticks of the PWM to be generated.

The last described module of the proposed solution is the ResponseComposer module. This module is used for sending all response UDP frames to PC through the hardware implemented OSI stack (Ethernet/IPv4/UDP) described some pages above. It sends back to PC the responses from various commands received also via UDP and also it sends to PC the signals' data acquired directly from Digital Control Module or from the DDR3 memory after an acquisition were performed and data was stored into the RAM memory.

All the modules and submodules of the proposed solution were written in Verilog HDL (Hardware Description Language). The leaf modules, like the CommandTranslator, the RegistersFile, the ResponseComposer, the TimestampManager, the DIO Record Sender, the DDR3 Operations Control Module, etc. are written as behavioural hardware description (finite state machines) and the composed modules like the Digital Control Module and the Memory Control Module are described as structural modules (instantiations of leaf modules and IPs). Leaf modules and composed modules were developed and simulated using AMD Vivado 2023.2, the Verilog variant is Verilog 2001 and the simulator used is Vivado Simulator. The bitstream embodying the whole behavioural and structural implementation generated by Vivado is programmed through development board's JTAG circuitry into the FPGA's attached 16 MB Flash Memory.

5. Tests and Results

In order to test and validate the functionalities of the proposed solution and to prove that it manages to obtain better performances than the

microcontrollers presented into the available market solutions section; the following criteria were taken into consideration:

- Time duration needed for initiation the transfer to PC of an Ethernet frame containing the changes is signal states together with the timestamp of the change occurrence.
- Number of PWM cycles elapsed between the moment a PWM input channels had changed its parameters and the moment the solution exposes the new measurement data for the acquired PWM signal.
- The propagation delay of the output channel reflecting the result of a multi-channel logical operation over between 2 and 31 inputs.

In order to measure the time needed for initiating an Ethernet transmission containing changed statuses of modified digital IOs the following setup (Fig. 9):

- The DUT (Device Under Test): the proposed solution, FpgaDaqEth, working at 100 MHz deployed on AMD Artix-7 XC7A100T on Digilent Arty A7-100T development board.
- The Device for Compare is an AMD MicroBlaze CPU configured to work at 200 MHz and to expose Ethernet communication.
- The Stimuli Device is SigWavy_U, the UART controlled variant of the PWM generation unit described in paper (Popovici *et al*., 2023).
- The Measurement Tool is Saleae Logic Pro 8, Logical Analyzer and Protocol Decoder described some pages ago as state of the art for digital signal acquisition
- The Configuration and Monitor Tool is the PC, which is connected by Ethernet to the proposed solution, FpgaDaqEth and to the device for compare, MicroBlaze core and by USB to the measurement tool.

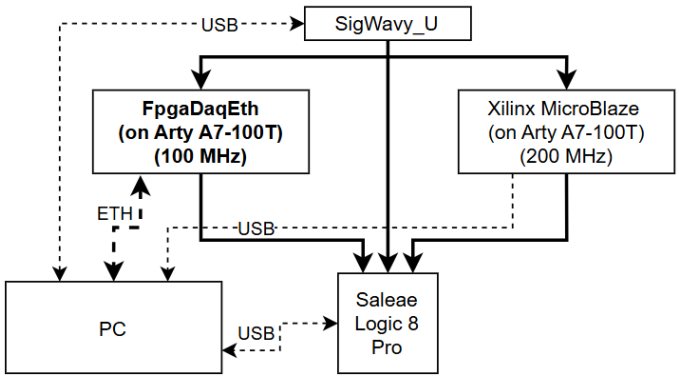


Fig. 9 – Test Setup for measuring initiation of Ethernet transmission containing IOs changes.

The proposed solution, FpgaDaqEth (operating at 100 MHz), is able to start an Ethernet frame transmission containing modified Digital IOs changes 19.31x times faster than an AMD MicroBlaze CPU running at 200 MHz deployed on the same FPGA, AMD Artix-7 (Fig. 10).

Delays between signal changes and Ethernet transmission initiations (µS)	
MicroBlaze on Arty A7-100T at 200 MHz	FpgaDaqEth on Arty A7-100T at 100 MHz
20.472	1.06

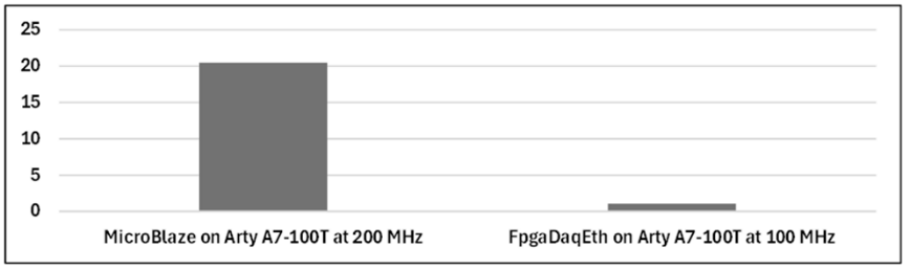


Fig. 10 – Time comparison between two devices for initiating Ethernet packets transmission.

The FPGA resources utilization comparison between the proposed design and the MicroBlaze core is presented in Fig. 11. It can be observed that the percentages of LUTs (look-up tables) are approximately equal. Both designs are using also the same number of BUFGs (global clock simple buffers) and approximately equal MMCM (mixed-mode clock manager) resources because the same number of clock signals are generated using the 100 MHz input clock: the main clock of the system and the 200 MHz reference clock for DDR3 MIG. The proposed solution uses almost 6x times lower BRAM (Block RAM) resources because it is a dedicated non-CPU solution, and it doesn't need local data and program memories for instructions execution. The increased IOs utilization of the proposed solution is the consequence of exposing multiple debug signals in the development phase of the project.

Device \ Used Resource Type	LUT	LUTRAM	FF	BRAM	IO	BUFG	MMCM
MicroBlaze for Digital Acquisition (200 MHz)	24%	1%	12%	57%	27%	13%	20%
FpgaDaqEth (100 MHz)	23%	2%	15%	10%	49%	13%	17%

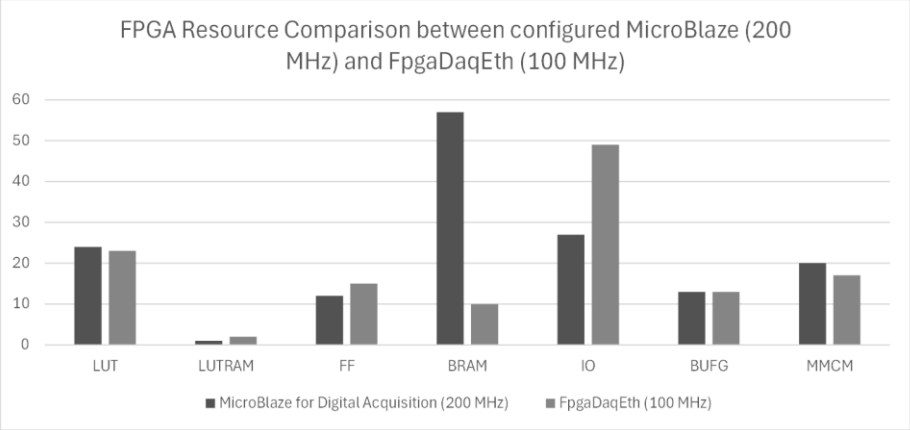


Fig. 11 – FPGA resources utilization – comparison between FpgaDaqEth (100 MHz) and MicroBlaze core (200 MHz).

For comparing the number of PWM cycles needed for detecting changes in a PWM signal, the setup also included the SigWavy_U PWM generation unit as Stimuli Device and the PC as configuration and monitor tool. The same measurement tool, Saleae Logic Pro 8, was used for performing signals recording and time duration measurements (Fig. 12). The device compared to the proposed solution is the Cortex-M7 running at 600 MHz described in the third section.

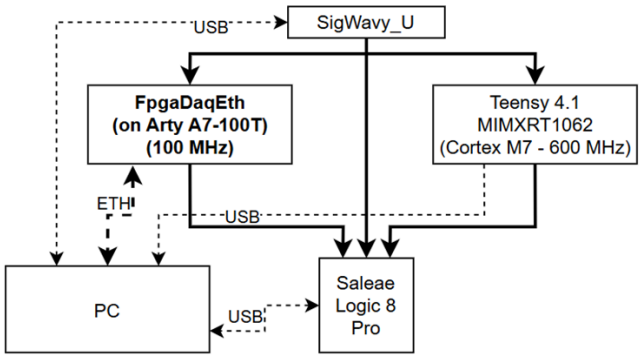


Fig. 12 – Test Setup for measuring and comparing the PWM cycles needed for detecting signals' parameters modifications.

The proposed solution, FpgaDaqEth, manages to detect and measure the changes in PWM signals after just one cycle of the modified signal, in comparison

with NXP MIMXRT1062, which operates at 600 MHz, and detects changes in between 2 and 5 cycles of the new signal. These measurements were performed using PWM frequencies between 20 and 100 Hz, 100 and 900 Hz, 1 and 9 KHz, 10 and 70 KHz (Fig. 13).

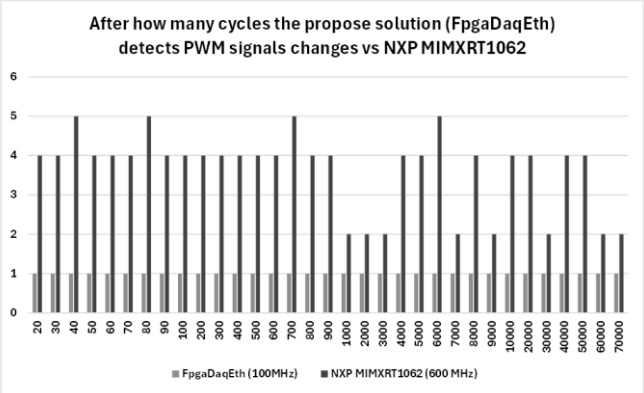


Fig. 13 – Comparison between proposed solution and the Cortex-M7 in terms of PWM cycles passed before measurement of the PWM's parameters changes are updated.

The last performance comparison was made between the proposed solution, FpgaDaqEth, running at 100 MHz, the Cortex-M4 running at 120 MHz and the Cortex-M7 running at 600 MHz. The test refers to measuring the time needed for performing a multi-channel logical operation on 2-31 inputs and outputting the result. Same PWM generation unit was used as stimuli device and same measurement tool was used for measuring the propagation delay of the multi-channel logic operation performed by compared devices (Fig. 14).

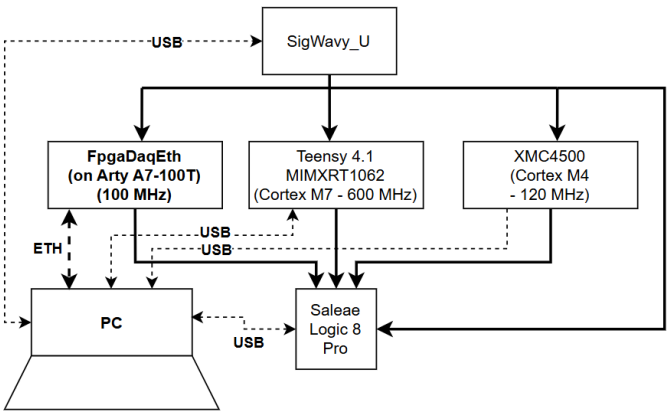
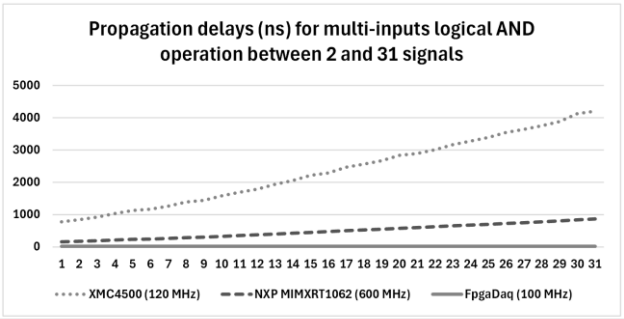


Fig. 14 – Test Setup for measuring and comparing propagation delays for multi-channel logical operations.

The proposed solution, operating at 100 MHz, manages to output the AND operation performed on between 2-31 input signals between 8x and 43.3x times faster than NXP MIMXRT1062 operating at 600 MHz. For the proposed solution the propagation delay is constant, 20 ns, and it is not influenced by the number of inputs (Fig. 15).



Logical AND	XMC4500 (120 MHz)	NXP MIMXRT1062 (600 MHz)	FpgaDaq (100 MHz)
2 signals	776 ns	160 ns	20 ns
31 signals	4200 ns	866 ns	20 ns

Fig. 15 – Comparison between proposed solution, Cortex-M4 and Cortex-M7 in terms of propagation delay for outputting the multi-channel logic operation AND for 2-31 inputs.

6. Transforming FpgaDaqEth into a RISC-V extension/hardware accelerator

Nowadays, the most popular processor architecture for integrating extensions and accelerators is RISC-V (Hennessy and Patterson, 2017), due to its open and facile to extend ISA (instruction set architecture). The categories of extensions designed for RISC-V include extensions for Vector/SIMD instructions, such as (Tagliavini *et al*., 2019), for cryptography algorithms accelerations, such as (Hoang *et al*., 2020), for artificial intelligence and machine learning, such as (Vreča *et al*., 2020), for networking and high-speed communications, such as (Tourres *et al*., 2021), for protected and optimized memory operations, such as (Ankit *et al*., 2020) and for digital signal processing, like (Bailey *et al*., 2018).

The fact that RISC-V extensions, accelerators, and coprocessors like the ones presented in papers (Bailey *et al*., 2018; Hung *et al*., 2021) and (Arnaud *et al*., 2020) have proven performances higher than classical general-purpose DSPs and some of them are already produced ASICs (application-specific integrated circuits) and the fact that the solution proposed in this paper has proven better performances than some general purpose microcontrollers, authors came to the conclusion that FpgaDaqEth can be adapted to be integrated as an

extension/hardware accelerator into a RISC-V core and this will represent the subject of a future project.

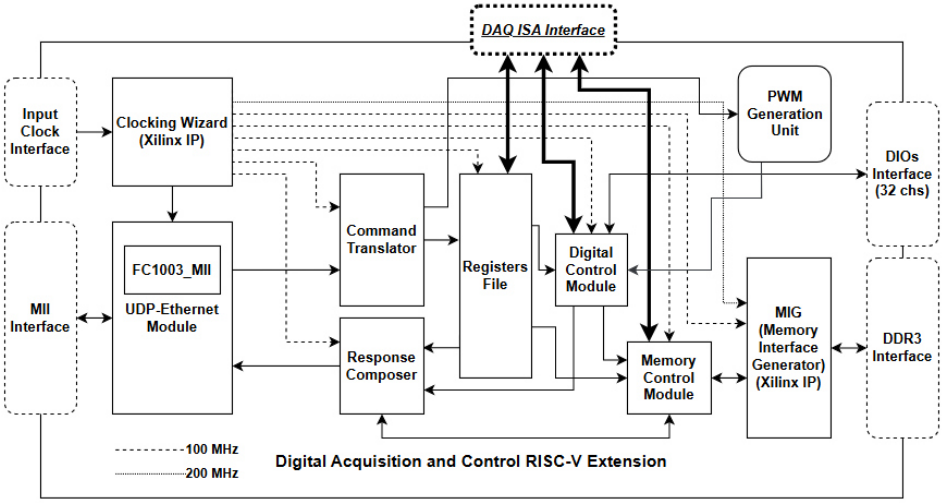


Fig. 16 – Proposed architecture for transforming the digital acquisition system into RISC-V digital acquisition extension.

The proposed architecture for transforming the digital acquisition system into a RISC-V extension is illustrated in Fig. 16. The most important architectural detail is the introduction of the DAQ ISA Interface, the interface which translates the fields of RISC-V ISA extensions into signals and data buses for controlling the RegistersFile module, the Digital Control Module and Memory Control Module. For debugging purposes and for offering flexibility in using the solution, the Ethernet control of the system won't be removed.

Table 4

Format of the RISC-V R-type instruction

R-type instruction					
funct7	rs2	rs1	funct3	rd	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

The ISA extension (Table 4) which contains all the instructions needed for controlling the solution will imitate the R-type instructions format (Table 4) - similar ISA extensions with the ones proposed by the authors in the papers (Popovici and Stan, 2022) and (Popovici *et al*., 2023) - and there will be instructions for:

- Setting configurations masks: direction, filter, negated inputs, output source, output values, multi-channel logical operation inputs selection, etc.
- Getting the configurations enumerated above, getting digital IOs statuses and PWM measurements.
- Starting/stopping the digital signal acquisition.
- Clearing the FPGA's attached RAM and resetting the extension.

Table 5

RISC-V ISA extension proposal for controlling digital acquisition extension (opcode of instructions will always be 1111010_2)

instr_name	funct7	rs1	rs2	rd
daqset_dios_dir	1000000	$_dios_directions_mask	xxxxx	xxxxx
daqset_dios_in_negs	1000001	$_dios_inputs_neg_mask	xxxxx	xxxxx
daqset_dios_in_filt	1000010	$_dios_inputs_filter_mask	xxxxx	xxxxx
daqset_dios_outs	1000011	$_dios_outputs_mask	xxxxx	xxxxx
daqset_dios_pwmouts	1000100	$_dios_outputs_pwmmask	xxxxx	xxxxx
daqset_dios_dirandfilt	1000101	$_dios_directions_mask	$_dios_inputs_filter_mask	xxxxx
daqset_dios_dirandnegs	1000110	$_dios_directions_mask	$_dios_inputs_neg_mask	xxxxx
daqset_dios_dirandouts	1000111	$_dios_directions_mask	$_dios_outputs_mask	xxxxx
daqset_multi_inp_and	1001000	$_dios_and_selection_mask	$_dios_and_out_dio_no	xxxxx
daqset_multi_inp_or	1001001	$_dios_or_selection_mask	$_dios_or_out_dio_no	xxxxx
daqset_multi_inp_xor	1001010	$_dios_xor_selection_mask	$_dios_xor_out_dio_no	xxxxx
daqget_dios_dir	1100000	xxxxx	xxxxx	$_dios_directions_mask
daqget_dios_in_negs	1100001	xxxxx	xxxxx	$_dios_inputs_neg_mask
daqget_dios_in_filt	1100010	xxxxx	xxxxx	$_dios_inputs_filter_mask
daqget_dios_outs	1100011	xxxxx	xxxxx	$_dios_outputs_mask
daqget_dios_pwmouts	1100100	xxxxx	xxxxx	$_dios_outputs_pwmmask
daqget_dios_values	1100101	xxxxx	xxxxx	$_dios_values
daqget_dios_pwms	1100110	$_dios_pwm_input_no	xxxxx	$_highTicks16_lowTicks16
daqrec_start_send_to_PC	1110000	xxxxx	xxxxx	$_32_bits_timestamp
daqrec_stop_send_to_PC	1110001	xxxxx	xxxxx	$_32_bits_timestamp
daqrec_start_store_in_RAM	1110010	xxxxx	xxxxx	$_32_bits_timestamp
daqrec_stop_store_in_RAM	1110011	xxxxx	xxxxx	$_32_bits_timestamp
daqrec_start_transf_to_PC	1110100	xxxxx	xxxxx	xxxxx
daqctrl_reset_DDR3	1111000	xxxxx	xxxxx	xxxxx
daqctrl_reset_system	1111001	xxxxx	xxxxx	xxxxx

The instructions from the ISA extension presented in Table 5 are using the R-type format from RISC-V specifications, in terms of data extracted from the fields and in terms of writing back requested values. The instructions' opcodes are the same for the digital acquisition extension and they are differentiated by the funct7 field. For some instructions, needed configurations are written in RISC-V registers and their values are extracted by the extended ISA instructions using source registers addresses. Other instructions are writing the requested data back to a destination register.

The most popular RISC-V implementations including the already produced chips embodying this architecture are using the pipelined datapath. The simplest datapath is the 5 stages pipeline proposed by (Hennessy and Patterson, 2017). As we can see in Table 6, all the instructions are performing actions in the IF (Instruction Fetch), ID (Instruction Decode) and EX (Execute Stages), they won't perform any action in MEM (Memory Access) and just a part of them will

write data to destination registers in the WB (Write Back) stage. In conclusion the instructions composing the extension ISA for digital acquisition and control will take 3 or 5 CPU cycles to be executed.

Table 6

The number of CPU cycles needed by each instruction of the proposed ISA extension

instr_name	IF	ID	EX	MEM	WB	Duration in CPU cycles
daqset_dios_dir	X	X	X			3
daqset_dios_in_negs	X	X	X			3
daqset_dios_in_filt	X	X	X			3
daqset_dios_outs	X	X	X			3
daqset_dios_pwmouts	X	X	X			3
daqset_dios_dirandfilt	X	X	X			3
daqset_dios_dirandnegs	X	X	X			3
daqset_dios_dirandouts	X	X	X			3
daqset_multi_inp_and	X	X	X			3
daqset_multi_inp_or	X	X	X			3
daqset_multi_inp_xor	X	X	X			3
daqget_dios_dir	X	X	X		X	5
daqget_dios_in_negs	X	X	X		X	5
daqget_dios_in_filt	X	X	X		X	5
daqget_dios_outs	X	X	X		X	5
daqget_dios_pwmouts	X	X	X		X	5
daqget_dios_values	X	X	X		X	5
daqget_dios_pwms	X	X	X		X	5
daqrec_start_send_to_PC	X	X	X		X	5
daqrec_stop_send_to_PC	X	X	X		X	5
daqrec_start_store_in_RAM	X	X	X		X	5
daqrec_stop_store_in_RAM	X	X	X		X	5
daqrec_start_transf_to_PC	X	X	X			3
daqctrl_reset_DDR3	X	X	X			3
daqctrl_reset_system	X	X	X			3

Because the instructions proposed for controlling the digital acquisition extension has the R-format like the arithmetic and logical instructions, a minimal number of changes in the RISC-V datapath architecture (Fig. 17) are necessary:

- the digital acquisition and control extension should be placed in the same pipeline stage with the ALU, the execution stage, because it takes values from the registers specified by source addresses and because some of the instructions are writing back results to destination register.
- the digital extension performs an action only when the specific opcode of the ISA extension and a valid funct7 field value are detected by the control signals generation block.
- The multiplexer selecting the data source for the destination register which usually takes values from ALU or from the output port of the data memory should be extended to take as input also the configurations or measurements requested to the digital acquisition extension.

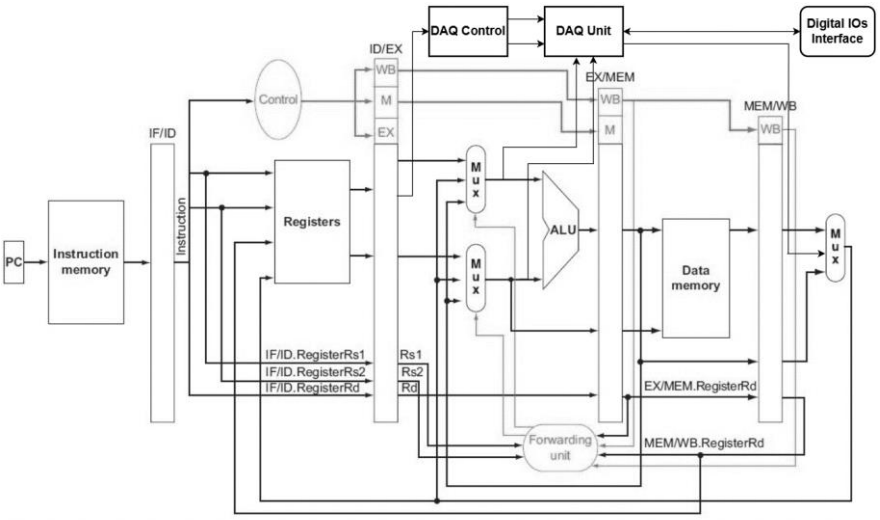


Fig. 17 – The RISC-V Datapath from (Hennessy and Patterson, 2017) modified by adding the Digital Acquisition Extension.

Last thing to consider in order to use the RISC-V extension for digital acquisition in real Embedded C applications is extending a compiler for generating the machine code specific to the instructions enumerated above. At the moment of writing this article, GCC is the only compiler to fully support RISC-V for C/C++ and Fortran and for OpenMP framework, for multiple variants: RV[32/64] with all subvariants and extensions I (integers), M (multiplying and division for integer extension), A (atomic operations), F and D (floating-point units for simple and double precision) and for almost all extensions described in the privileged and unprivileged RISC-V instruction specifications.

The first step in extending the GCC compiler for supporting the proposed ISA extension is adding the compile extension string which will be attached to the compiling option '-march=ISA-string' (machine architecture option) and implementing the support for this option. For example, if the extension option proposed by authors is 'daq', the whole compiling option for generating code for a 64-bit RISC-V core with integers support, with multiplication/division unit for integers, and with the proposed digital acquisition unit will be '-march=ISA-rv64imdaq'.

The second step is implementing the support for the proposed ISA extension controlling the digital acquisition extension. By reading the chapter of the GCC documentation describing the RISC-V built-in functions, authors came to the idea that built-in functions are the best way for the compiler to generate the machine code specific to the proposed ISA extension. The prototypes of the built-

in functions proposed for GCC to generate the ISA extension machine code are listed in Table 7.

Table 7

The prototypes of the built-in functions proposed to be used by GCC for generating the instructions of digital acquisition extension ISA

Instruction Name	Built-in function for generating the instruction machine code with GCC
daqset_dios_dir	`void __builtin_riscv_daq_set_dios_dir(uint32_t);`
daqset_dios_in_negs	`void __builtin_riscv_daq_set_dios_in_negs(uint32_t);`
daqset_dios_in_filt	`void __builtin_riscv_daq_set_dios_in_filt(uint32_t);`
daqset_dios_outs	`void __builtin_riscv_daq_set_dios_outs(uint32_t);`
daqset_dios_pwmouts	`void __builtin_riscv_daq_set_dios_pwmouts(uint32_t);`
daqset_dios_dirandfilt	`void __builtin_riscv_daq_set_dios_dir_and_filt(uint32_t, uint32_t);`
daqset_dios_dirandnegs	`void __builtin_riscv_daq_set_dios_dir_and_negs(uint32_t, uint32_t);`
daqset_dios_dirandouts	`void __builtin_riscv_daq_set_dios_dir_and_outs(uint32_t, uint32_t);`
daqset_multi_inp_and	`void __builtin_riscv_daq_set_multi_inp_and(uint32_t, uint32_t);`
daqset_multi_inp_or	`void __builtin_riscv_daq_set_multi_inp_or(uint32_t, uint32_t);`
daqset_multi_inp_xor	`void __builtin_riscv_daq_set_multi_inp_xor(uint32_t, uint32_t);`
daqget_dios_dir	`uint32_t __builtin_riscv_daq_get_dios_dir(void);`
daqget_dios_in_negs	`uint32_t __builtin_riscv_daq_get_dios_in_negs(void);`
daqget_dios_in_filt	`uint32_t __builtin_riscv_daq_get_dios_in_filt(void);`
daqget_dios_outs	`uint32_t __builtin_riscv_daq_get_dios_outs(void);`
daqget_dios_pwmouts	`uint32_t __builtin_riscv_daq_get_dios_pwmouts(void);`
daqget_dios_values	`uint32_t __builtin_riscv_daq_get_dios_values(void);`
daqget_dios_pwms	`uint32_t __builtin_riscv_daq_get_dios_pwms(uint32_t);`
daqrec_start_send_to_PC	`uint32_t __builtin_riscv_daq_rec_start_send_to_PC(void);`
daqrec_stop_send_to_PC	`uint32_t __builtin_riscv_daq_rec_stop_send_to_PC(void);`
daqrec_start_store_in_RAM	`uint32_t __builtin_riscv_daq_rec_start_store_in_RAM(void);`
daqrec_stop_store_in_RAM	`uint32_t __builtin_riscv_daq_rec_stop_store_in_RAM(void);`
daqrec_start_transf_to_PC	`uint32_t __builtin_riscv_daq_rec_start_transf_to_PC(void);`
daqctrl_reset_DDR3	`void __builtin_riscv_daq_ctrl_reset_DDR3(void);`
daqctrl_reset_system	`void __builtin_riscv_daq_ctrl_reset_system(void);`

7. Software Tools developed for controlling FpgaDaqEth

In order to configure the digital acquisition system from the PC connected to it, two software components were developed, the first one is called FpgaDaqEth Library, which is a Windows DLL (dynamic-linked library), and FpgaDaqEth GUI which is a Windows GUI (graphical user interface) using the functions exposed by the DLL.

Both components were developed as a single .NET Framework 4.7.2 project under Visual Studio 2022, named FpgaDaqEthSW. The project contains the following C# classes:

- UdpFrame: an instance of this class embodies the contents an UDP frame sent/received to/from the proposed solution and contains the direction of the packet (RX or TX) and the payload as a byte array.
- UdpDriver: the instance of this class represents a wrapper for the Socket class configured as UDP socket and containing specific methods such as send and receive and handling all the possible exceptions which may be thrown by the Socket class methods.
- FpgaDaqEthDriver: the instance of this class is a singleton object for the project and contains all fields and methods necessary for controlling

FpgaDaqEth; it instantiates and uses an UdpDriver instance for communicating via UDP to the proposed solution connected to the PC through 100 Mbps Ethernet.

- FpgaDaqEthGui: is the Windows Forms GUI (Fig. 18) which uses the FpgaDaqEthDriver singleton for enabling user to configure and use the proposed solution. The user can configure the directions of the 32 channels, the filter and negate masks for the inputs, the multi-channel operations AND, OR and XOR masks and the outputs source mask. Also, user can request the state of the channels at any time instance, to start/stop the acquisition to attached RAM or PC's RAM, to plot the acquired signals and to acquire PWM measurement of every channel.
- SignalCache: an instance of this class is used for reading all the data from the trace file containing the acquired signals data and for transforming this data into plotted signals.

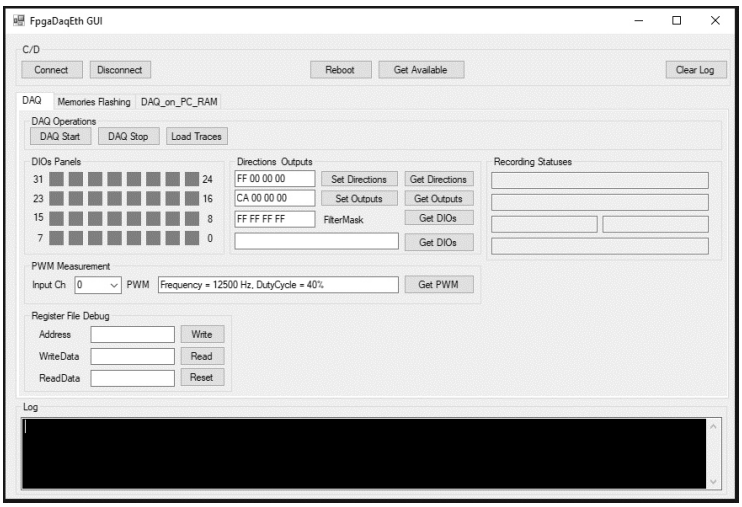


Fig. 18 – FpgaDaqEth Graphical User Interface screen shot.

The FpgaDaqEth GUI was used also for flashing new Bitfile on the FPGA deploying the solution and for debugging it by offering access to registers operations through the driver. After an acquisition cycle is performed user can launch a separated window for plotting the acquired signals (Fig. 19).

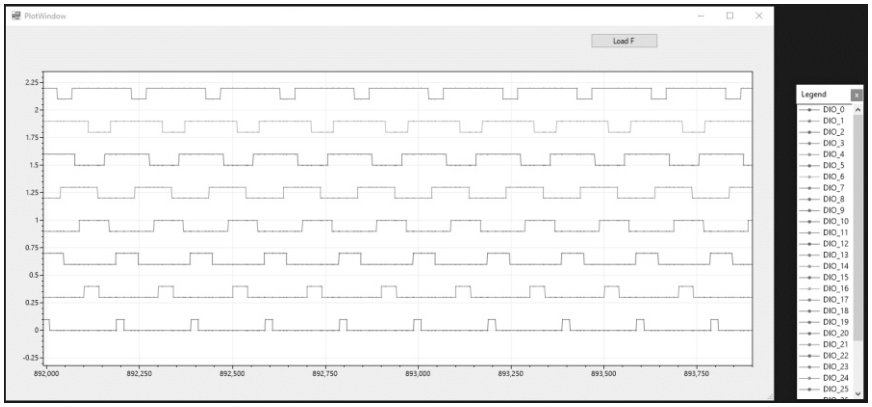


Fig. 19 – Plot Window GUI showing PWM signals acquired by the solution and transferred to PC from attached DDR3 RAM memory.

The signals data acquired and stored directly into the hosting PC's RAM can be visualised as table/chart view together with their timestamp into a dedicated tab page of the FpgaDaqEth GUI and will also be available for graphical plotting in a future update.

8. Conclusions

FpgaDaqEth, running at 100 MHz, is able to start an Ethernet packet transmission containing changed digital IOs changes 19.31x times faster than a AMD MicroBlaze CPU running at 200 MHz deployed on the same FPGA, AMD Artix-7 XC7A100T.

The proposed solution manages to detect and measure the changes in PWM signals after just one cycle of the modified signal, in comparison with the Cortex-M7, which operates at 600 MHz, and detects changes in between 2 and 5 cycles of the new signal.

The proposed solution, operating at 100 MHz, manages to output the AND operation performed on between 2-31 input signals between 8x and 43.3x times faster than the Cortex-M7 operating at 600 MHz. For the proposed solution, the propagation delay is constant, 20 ns, and it is not influenced by the number of digital inputs.

The increased time efficiency and the predictable behaviour are the benefits of the architectural choices presented in the sections above (Domain-Specific Architecture) and they are reasons for going forward with its integration into a RISC-V extension, this architecture being the most popular one for embedding hardware accelerators and ISA extensions.

REFERENCES

Ankit A., Chakraborty I., Agrawal A., Ali M., Roy K., *Circuits and Architectures for In-Memory Computing-Based Machine Learning Accelerators*, IEEE Micro, vol. 40, no. 6, pp. 8-22, 2020.

Arnaud A., Miguez M., Gak J., Puyol R., Garcia-Ramirez R., *A RISC-V Based Medical Implantable SoC for High Voltage and Current Tissue Stimulus*, in IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), San Jose, Costa Rica, 2020.

Bailey S., Han J., Rigge P., Lin R., Chang E., Mao H., *A Generated Multirate Signal Analysis RISC-V SoC in 16nm FinFET*, in IEEE Asian Solid-State Circuits Conference (A-SSCC), Tainan, Taiwan, 2018.

Droms R., *Dynamic Host Configuration Protocol*, Request for Comments: 2131, Network Working Group, Bucknell University, March 1997.

Gelfand I.M., Vilenkin N. Ya, *Generalized Functions: Applications of Harmonic Analysis*, Academic Press, 2014, ISBN: 9781483262246.

Hennessy J.L., Patterson D.A., *Computer Organization and Design. The Hardware/Software Interface. RISC-V Edition*, Morgan Kaufmann Publishers, 2017, ISBN: 9780128122754.

Hoang T.T., Duran C., Tsukamato A., Suzaki K., Pham C.K., *Cryptographic Accelerators for Trusted Execution Environment in RISC-V Processors*, in IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 2020.

Holton T., *Digital Signal Processing: Principles and Applications*, Cambridge University Press, 2021, ISBN: 9781108418447.

Hornig C, *A Standard for the Transmission of IP Datagrams over Ethernet Networks*, Request for Comments: 894, Network Working Group, Symbolics Cambridge Research Center, April 1984.

Hung Y., Chang Y., Lee S., *An Energy-efficient and Programmable RISC-V CNN Coprocessor for Real-time Epilepsy Detection and Identification on Wearable Devices*, in IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2021.

Plummer D.C., *An Ethernet Address Resolution Protocol*, Request for Comments: 826, MIT, November 1982.

Popovici C.A., Stan A., *Extending a RISC-V Core with a CAN-FD Communication Unit*, in 2022 26th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2022, https://doi.org/10.1109/ICSTCC55426.2022.9931880 .

Popovici C.A., Stan A., *Real-Time RISC-V-Based CAN-FD Bus Diagnosis Tool*, Micromachines, vol. 14, no. 1, p. 196, 2023, https://doi.org/10.3390/mi14010196 .

Popovici C.A., Stan A., Manta V.I., *RISC-V Extension for Optimized PWM Control*, in 27th International Conference on System Theory, Control and Computing (ICSTCC), Timișoara, Romania, 2023, https://doi.org/10.1109/ICSTCC59206.2023.10308510 .

Postel J., *Internet Control Message Protocol*, Request for Comments: 792, Information Sciences Institute University of Southern California, September 1981.

Postel J., *Internet Protocol*, Request for Comments: 791, Information Sciences Institute University of Southern California, September 1981.

Postel J., *User Datagram Protocol*, Request for Comments: 768, Information Sciences Institute University of Southern California, September 1980.

Proakis J.G., Manolakis D.G., *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice-Hall International, Third Edition, 1995, ISBN: 0133943389.

Shannon C.E., *Communication in the Presence of Noice*, Proceedings of the IRE, Volume 27, Issue 1, January 1949.

Tagliavini G., Mach S., Rossi D., Marongiu A., Benini L., *Design and Evaluation of Small Float SIMD extensions to the RISC-V ISA*, in Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019.

Tourres M., Chavet C., Le Gal B., Crenne J., Coussy P., *Extended RISC-V hardware architecture for future digital communication systems*, in IEEE 4th 5G World Forum (5GWF), Montreal, QC, Canada, 2021.

Vreča J., Sturm K., Gungl E., Merchant F., *Accelerating Deep Learning Inference in Constrained Embedded Devices Using Hardware Loops and a Dot Product Unit*, IEEE Access, vol. 8, pp. 165913-165926, 2020.

SISTEM DE ACHIZIȚIE A SEMNALELOR DIGITALE BAZAT PE FPGA CA SOLUȚIE SPECIFICĂ DOMENIULUI DE UTILIZARE PENTRU INTEGRARE ÎN ACCELERATOARE HARDWARE

(Rezumat)

Domenii precum IoT (Internetul lucrurilor), ADAS (Sisteme avansate de asistentă a șoferului), Casele Inteligentele, Orașele Inteligente si XaaS (Totul ca un serviciu) au pus o mare presiune în ultimii cincisprezece ani pe inginerii din domeniul calculatoarelor pentru a proiecta procesoare capabile de calcul de înaltă performanță și precizie, efectuat în vederea interconectării în rețele de comunicații de mare viteză și consumând mai puțină putere electrică în scopul alimentării lor de la baterii sau de la surse de energie regenerabile.

De aproape două decenii, legile clasice ale VLSI (circuite cu grad foarte mare de integrare) precum cele ale lui Moore și Dennard nu mai pot fi aplicate pentru a introduce de două ori mai mulți tranzistori în aceeași arie de siliciu la fiecare doi ani, păstrând în același timp densitatea de energie constantă. Cea mai fezabilă soluție este utilizarea de DSA (arhitecturi specifice domeniilor de utilizare) precum acceleratoarele hardware specifice unor domenii, care înlocuiesc necesitatea implementării în manieră programatică a algoritmilor specifici unui domeniu prin implementarea hardware la nivel de circuit digital a acelorași algoritmi în paralel cu programele software dezvoltate pentru procesoarele clasice.

Un domeniu al electronicii care beneficiază de utilizarea arhitecturilor specifice domeniilor de utilizare în detrimentul microcontrolerelor de uz general este procesarea digital a semnalelor (DSP). Proiectarea circuitelor digitale sau a sistemelor digitale

complexe nu ar fi posibilă în prezent fără utilizarea analizoarelor logice, ale decodoarelor de protocoale de comunicație, ale osciloscoapelor digitale și ale recorderelor digitale de semnale. Aceste unelte sunt folosite în procesele de prototipare și verificare ale circuitelor și sistemelor digitale pentru înregistrarea, afișarea, decodarea și analiza semnalelor interne și externe.

Acest articol propune un exemplu de arhitectură specifică domeniului de utilizare, un sistem de achiziție și control a semnalelor digitale numit FpgaDaqEth, care înregistrează date de la 32 de canale digitale cu o frecvență de eșantionare de 100 MS/s, care operează pe un FPGA tactat la 100 MHz, utilizând o memorie RAM DDR3 și o comunicație Ethernet de 100 Mbps cu calculatorul gazdă ce rulează aplicația cu interfață grafică. Acest proiect oferă o întârziere de propagare constantă a rezultatului unei operații logice efectuate pe mai multe semnale (2 - 31), de 20 ns, între 8 și 43,34 ori mai rapid decât în cazul utilizării unui microcontroler care funcționează la o frecvență de 6 ori mai mare. De asemenea, soluția propusă inițiază trimiterea unui pachet UDP conținând modificările stărilor semnalelor de intrare și momentul modificării lor de 19,31 ori mai rapid decât un procesor AMD MicroBlaze ce funcționează la o frecvență de 2 ori mai mare decât proiectul de față. Soluția poate detecta modificările produse în parametrii unui semnal PWM de intrare după eșantionarea a doar o singură perioadă din semnalul modificat.

Proiectul propus funcționează pe un FPGA AMD XC7A100T ca soluție de sine stătătoare și poate fi integrat ca o extensie de procesor RISC-V într-un proiect viitor.