Praca wykonana w ramach projektu TEAM kierowanego przez Prof. Pawła Moskala. Projekt finansowany przez Fundację na rzecz Nauki Polskiej.

Uniwersytet Jagielloński w Krakowie

Wydział Fizyki, Astronomii i Informatyki Stosowanej

Karol Farbaniec

Nr albumu: 1115092

Tomographic data processing and visualization on programmable devices

Praca magisterska

na kierunku Informatyka Stosowana spec. Ogólna

Praca wykonana pod kierunkiem

Dr. Grzegorza Korcyla

Zespół Zakładów Informatyki Stosowanej

Kraków 2019

This work is supported by the Foundation for Polish Science under Grant TEAM/2017-4/39

Contents

1. Introduction

Technological advances bring innovation in nowadays evidence-based medical examination among others by means of computer science related research areas. Imaging of interior of living systems can provide information on its structure, anatomy and functional information. Magnetic resonance imaging (MRI) and Computed Tomography (CT) provide excellent details on structure and anatomy of organs. Nuclear medicine techniques complement the insight into human body by providing functional information like metabolic processes. Positron Emission Tomography (PET) has introduced a new aspect to existing approaches.

The aim of this work is to implement selected data processing stages for Modular J-PET detector readout and explore feasibility of hardware acceleration of image reconstruction. Moreover, this work covers main aspects of FPGA design process.

1.1. PET Tomography

Imaging technique creates visual representation of interior by means of analysing specific physical phenomena. One of the most common type of imaging is tomography, where visualization is constructed by combining together cross sections of examined object.

There are many categories of tomography depending on used physical phenomena and they are deployed in wide range of medical, scientific and industrial applications. Some tomography types worth to mention are:

- Positron Emission Tomography, where positron decays are used for imaging purposes. Commonly applied in medical imaging [1].
- Computed Tomography (CT), where gamma rays are used for creating cross sections. Widely used in industrial and medical applications [2].
- Optical Coherence Tomography (OCT), where optical signals are transmitted through or reflected by a tissue, these signals are used to reconstruct spatial image of tissue. Applied in ophthalmology [3].
- Atom-Probe Tomography, where mass-to-charge ratio is computed for atoms using electric field in complex set-up. This method is used in material sciences for examination of material composition [4].

- Nuclear Magnetic Resonance tomography (NMR), where resonance induced by magnetic field is used for spatial imaging. Most often found in medical applications [5].

Basically, in Positron Emission Tomography radiopharmaceutical substance is injected into patients' body. Emitted positrons annihilate with electrons and produce two gamma quanta. The substance is distributed all over human body through blood and gets metabolized by organs. Depending on type of cells the level of metabolism differs. Cancerous cells have typically much higher level therefore they absorb more of the substance. This in turn results in more gamma pairs emitted from a particular location. Important feature of the gamma pair is that they are released in opposite direction in respect to each other. That sort of events is fundamental for imaging purposes. Except emission in opposite direction other phenomena are present, so one of challenges is to properly classify and filter gathered experimental data. Valid type of events creates Line-Of-Response (LOR) [Figure 1]. Large dataset of LORs that is appropriate to be analysed with statistical methods is primary source of information for imaging algorithms. Raw geometrical information on classified events is not enough for nowadays positron emission medical imaging setups. Except geometry it is crucial to precisely measure time of events. That sort of timing information adds advanced filtering and classifying prospects and is effectively used in imaging with PET facilities of Time of Flight (TOF) type. To define TOF it is worth to introduce coincidence as two gamma quanta hitting opposite detectors within short time distance. They can be considered as a single decay event data. Based on that time difference measurement one can calculate location of each annihilation event. If perfect TOF resolution could be achieved, then there would be no need for reconstruction, because exact location would be derived based on analysed phenomenon. Time of Flight measurement imperfection infers quality of the reconstructed image. About 12 cm uncertainty in annihilation location is induced by time difference resolution of around one nanosecond. That regions of possible event location are described as Region-of-Response (ROR) [Figure 1]. Moreover, TOF data provides information on detector quality, readout electronics and based on that proper calibration can be developed.

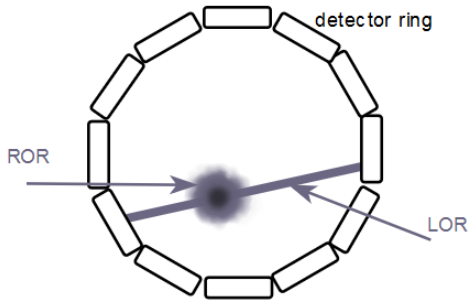

Figure 1 LOR and ROR scheme on detector ring

1.2. Data acquisition systems

Since early beginnings of experimental sciences, the process of observation was crucial in conducting research, just after proper design of experiment. With development of science and technology it became possible to automate monitoring of physical phenomena. Advances in electronics and computer science enabled appropriate form of gathering, processing and archiving certain information from physical signals, which states in the genesis of Data Acquisition System (DAQ). Nowadays, sensing subsystem, its transition to digital domain, processing and storage units combined together are described as DAQ.

Design of feasible DAQ is considered in reference to its application requirements derived from nature of measured signals. For applications where signals and systems do not require advanced, customized control and data processing, it is commercially viable to use off-the-shelf solutions. However wide range of products and experimental setups must have customized hardware and software solutions because of:

- high-level of specialization and complexity in data acquisition process,
- cost, energy and performance optimizations

One of the most representative branches of science where state-of-the-art technologies in DAQs are those commonly used is experimental particle physics. The reason of that is the need for customized setup often dedicated to particular experiment [6].

Tomographic data processing and visualization system components can be classified as subset of High Energy Physics DAQ elements. Therefore, description of assumptions and essential techniques will be presented in terms of particle physics experiment application. Specialized domain of application infers wide range of definitions and methods that are unique for experimental particle physics.

Typically, high performance DAQ systems in scientific applications operate in complex environment interacting with experimenters, sensors, external control equipment, machines and data archiving subsystems. In all these interrelations data path from detector through processing for later storage and analysis is core element existing in majority of systems [Figure 2].

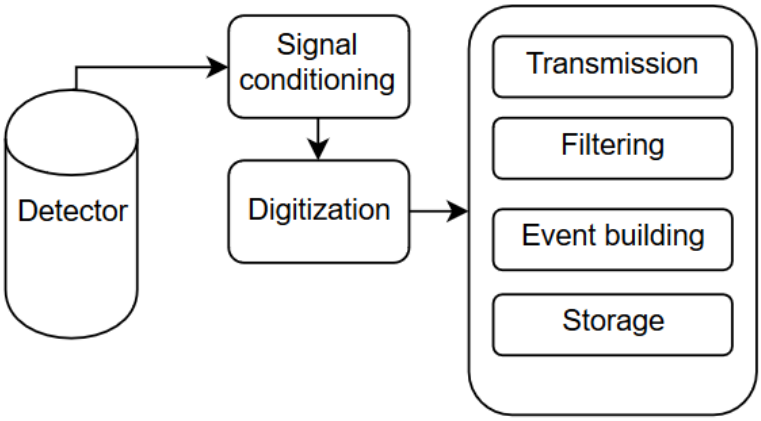

Figure 2 High Energy Physics DAQ conceptual scheme

Detectors are data sources for later stages of acquisition and processing. Main characteristics of detector in reference to DAQ design are:

- Detector response: In general, considered as an electrical response. Connected with deposited energy by received particle.
- Detector response function: Characterizes pulses for different types of radiation.
- Response time: It is the time detector needs to form the signal after radiation. During this period detector is insensitive, it cannot accept another event. This contributes to dead time of detector.
- Dead time: Finite time it takes for whole detecting system to process event. During this period detector can be insensitive, then data is lost. On the other hand for sensitive detector distortion of signal arises and data from both events is corrupted.
- Detector efficiency: Overall benchmark that relates registered events to events emitted by radiation source.

Detector characteristics have to be taken into account while developing its readout system. Next stage is Front-End Electronics (FEE). Pulses from detector are there conditioned and shaped for later interfacing with digitizing stage.

Transformation from analog to digital domain in reference to measurement of detector signals characteristics can be classified in two categories. First is the precise time measurement of the moment when a digital signal switches its logical state. Components performing that kind of operations on signals are called Time-to-Digital Converters (TDC). In deliberations on time to digital conversion one can differentiate two types. First is based on delay chain and single clocking signal. When input signal arises, then its propagated through connected electronic elements with known propagation delays (i.e. registers) and on a clock event, values from all elements are sampled. Results of sampling are decoded with thermometer code decoder (N lower ‘1 in bit vector corresponds to value of N) and normalized. Second

approach utilizes multiphase clocking signals. Multiple clocks with various frequencies and phase shifts sample the input signal with well-defined intervals. With known frequency of each clock and time dependencies between them one can calculate duration of signal and then decode final results. After time measurement, acquired data is packed and sent to the next stage of processing. Other approach in digitization is represented by Analog to Digital Converters (ADC), which sample the analog signal with fixed time interval, the way oscilloscope does.

Afterwards the digital data is pre-processed (i.e. parsed to proper data format) and filtered for data reduction. Then, the data is transmitted between system components, because some operations are executed on specific platforms. Finally, the results are processed, packed in data units and transmitted for Event Building, which assembles data fragments from the entire system and forwards to storage devices.

1.3. Image reconstruction

Positron decays in PET tomography occurs in stochastic manner, so they can be considered in terms of projections of examined object from different angles. Basically, projections in image reconstruction are considered as integral for fixed orientation angles as seen on scheme [Figure 3].

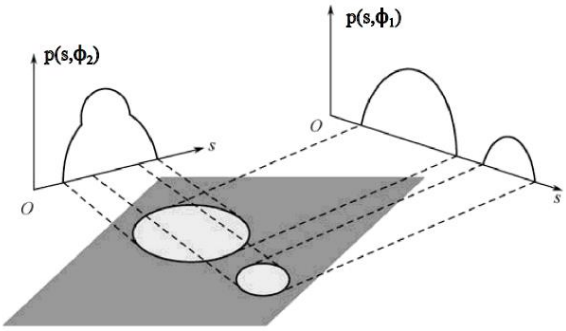

Figure 3 Projections from two angles [7]

The more different projection acquired, the better image reconstruction can be obtained. It is possible to simply reconstruct image by LORs histogramization. On areas with more radiation activity, more LORs will appear. That simplified approach results in image that is diffused, but there is an analytical way to improve reconstructed image quality by filtering.

In image reconstruction transformation that formalize projections calculation is called Radon Transform defined with equation (1).

$$\mathrm{p(s,\phi)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\cdot\delta(x\cdot cos(\phi)+y\cdot sin(\phi)-\mathrm{s})\,dxdy \quad (1)$$

The function p is defined on the space of straight lines $L(s,\phi)$ in R^2 by the linear integral along each line. In PET tomography Radon Transform discrete implementations take into account only part of the plane with LORs bounded by detecting system. In reference to Radon Transform each LOR corresponds to a unique pair of parameters $p(s,\phi)$ that matches its line [8]. In discrete case sampling infers data quantization, so angles ϕ and rays s have limited amount of possible values. All pairs of discrete

$$\left(s_i,\phi_{\mathrm{j}}\right)\ where\ i\ \in\langle 0,N-1\rangle\ and\ \ j\in\langle 0,M-1\rangle, \quad (2)$$

are combined in 2D histogram, where N and M stands for bins count on ray axis and angle axis. Two dimensional histogram with Discrete Radon Transform result is known as sinogram, that naming is connected with nature of transform results. In example, for point source Radon Transform results in sinewave.

Sinogram construction is the first stage in Filtered Backprojection (FBP), which is a common image reconstruction method studied in this work. Following that in FBP filtering is performed in frequency domain to reduce blur. Having regard to project-slice theorem [9] projection in frequency domain can be obtained by 1D Fourier Transform (FT) for each ϕ value, where a ray is treated as variable. In this method, implementations of Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) are calculated. Usually DFT and IDFT processing is realized by Fast Fourier Transform (FFT) algorithm. After that filtering is performed.

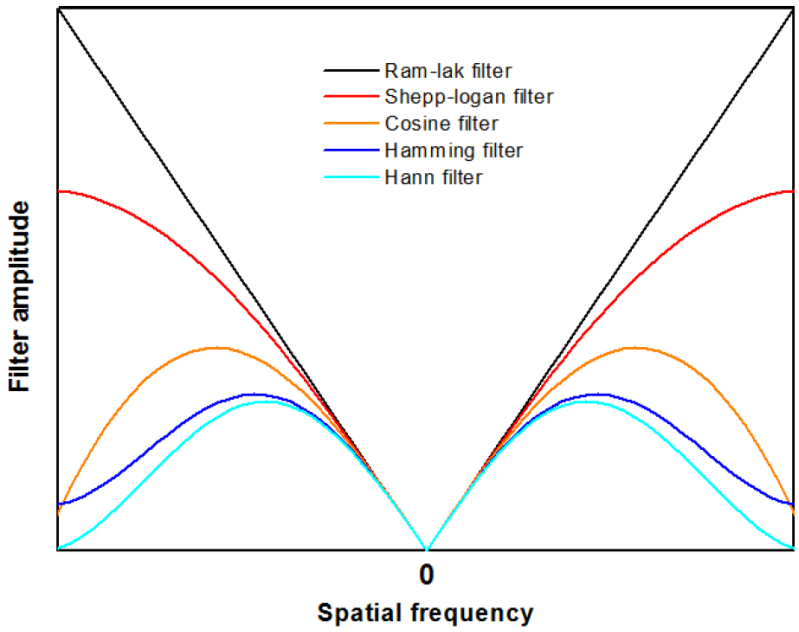

Figure 4 Filter characteristics [10]

Basic filter in reference to FBP image reconstruction is Ram-Lak filter also called Ramp Filter, where frequency spectrum from each 1D FT result is scaled linearly. Low frequencies are throttled and high frequencies are amplified. Based on Ramp Filter few others are derived [10], with some attenuation in highest frequencies. In example, Shepp-Logan Filter is obtained by multiplying Ramp Filter by Sinc function and similarly Hamming Filter is obtained by multiplying Ramp Filter with Hamming Window [Figure 4]. After filtering Inverse Fourier Transform (IFT) is performed. Filtering reduces background noise by attenuating low frequencies. Sharpening edges and image contrast extraction is obtained by passing high frequencies.

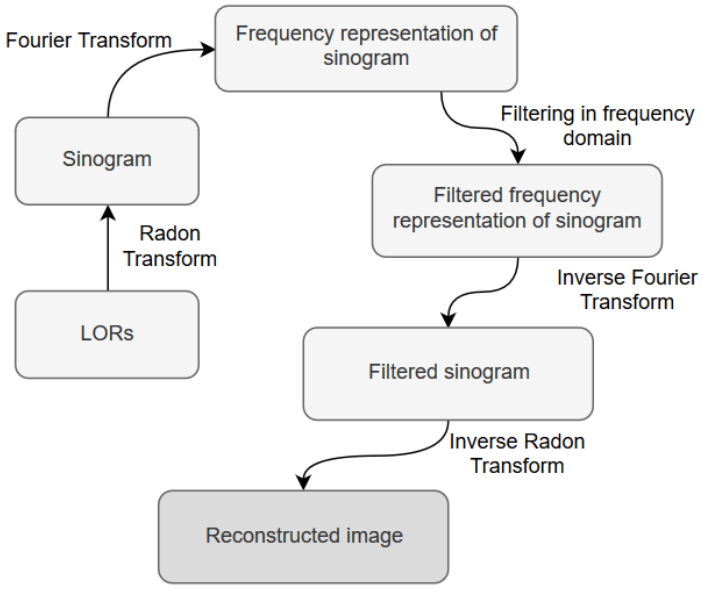

Figure 5 Filtered Backprojection method scheme

As a last step in image reconstruction filtered sinogram is back projected with Inverse Radon Transform [Figure 5]. Time of Flight information supplements projections from LORs. LORs are projected with higher contribution from estimated place of emission.

Other common methods are based on modelling detection system with knowledge of detector and physics process. LORs are treated there as input data in iterative process of improving modelled system. Common and representative method of that algorithms family is Maximum Likelihood Expectation Maximization (MLEM) method [11].

2. FPGA technology in heterogenous computing

2.1. Electronic aspects of FPGA Design

FPGA devices as other digital integrated circuits (IC) are built with analog components, which has impact on design process. Logical signals are represented in electronics as proper band values of voltage. Bands are set depending on standard, typically voltage level over 90% of supply voltage stands for logical “1” and voltage below 10% for logical “0” [12]. FPGAs are fabricated in Complementary Metal-Oxide Semiconductor (CMOS) technology, which enables power efficiency and very large scale of integration (VLSI) [12]. Essential component from which more complex elements are built is a transistor [Figure 6] which acts like a switch.

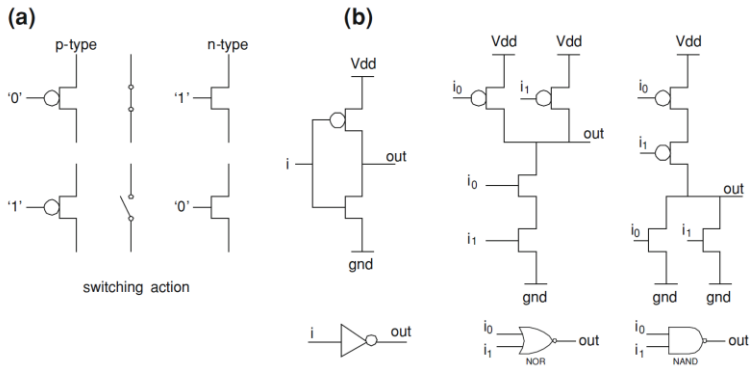

Figure 6 a) transistor symbols, b) example logic gates from transistors
source: J.-P. Deschamps, G. D. Sutter and E. Cantó, Guide to FPGA Implementation of Arithmetic Functions, Springer, 2012

Basic characteristic of electronic component is propagation delay depending on supply voltage and temperature. To precisely determine propagation timing there is a distinction between transition from low-to-high and from high-to-low levels called rise time and fall time respectively [Figure 7].

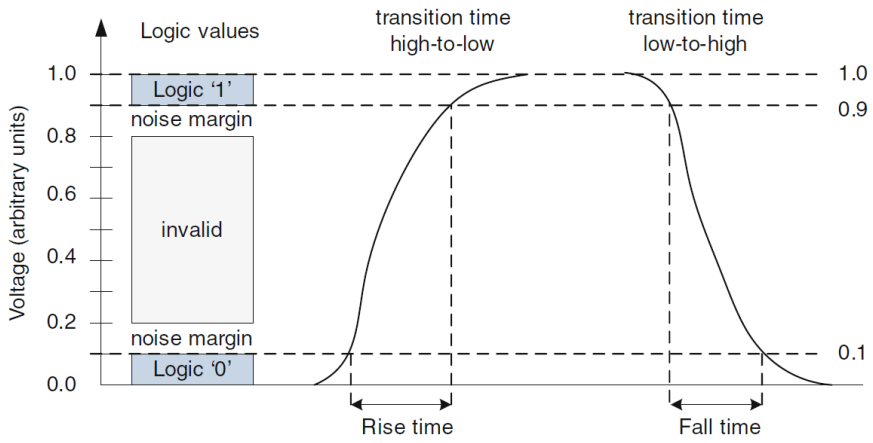

Figure 7 Example transition timing scheme

source: J.-P. Deschamps, G. D. Sutter and E. Cantó, Guide to FPGA Implementation of Arithmetic Functions, Springer, 2012

Majority of FPGA designs is synchronous, that means electronic modules are clocked, in other words edge-sensitive for clocking signals. Value in a component is sampled only on clock event (rising or falling edge), and should be distributed in proper way, that enables synchronous execution of different parts of design. Clocking is a crucial factor in synchronous designs and it causes few aspects for consideration.

Basic storage element in electronic circuit is a register. One can distinguish two types of registers: first are edge sensitive flip-flops, second are level sensitive latches. For flip-flops data on its input is sampled only on clock event (rising or falling edge) and passed to output. On the other hand, latches are level sensitive, and theirs output can change in response to something other than clock signal. Latches are not widely used in FPGA technology, because many FPGA vendors do not support its direct hardware implementation and attempts to synthesize them could result in a different circuit with not predictable behaviour. Flip-flops are common basic form of data storage, in today's FPGAs D-type flip-flops are used (abbr. D-FF) [12].

Edge sensitive logic operate properly only when data on its inputs is stable some time before and after clock event. Setup time is minimum amount of time for input data to be held stable before clock event. Hold time is minimum amount of time for input data to be held stable after clock event. It is also worth to define propagation delay as the amount of time after clock event when output of flip-flop becomes stable.

Main vulnerabilities in reference to electronic foundations of clocked designs in FPGA technology are [12]:

- Asynchronous signals
- Clock domain crossing

Other failures in FPGA technology descended from electronic domain are [12]:

- Metastability is present when design has setup or hold timing violations. That means logic could enter quasi-stable state. In that case output is unpredictable and it is considered as failure of design. In the end flip-flop settles down in high or low level.
- Glitch is electrical pulse of short duration, usually unwanted. In general produced by imbalanced internal interconnection delays.
- Spike is electrical, short pulse similar to glitch caused by unwanted oscillations of a signal or undesired inductive or capacitive coupling.

In programmable logic devices except interfacing and communication resources like transceivers, converters and input/output buffers it is necessary to describe resources for computing purposes. Essentially, in programmable technologies every device is built out of Configurable Logic Blocks (abbr. CLB). CLBs differs in their internal structure depending on device family and they are main resource for implementing general purpose combinatorial or sequential logic [13]. As a case study Xilinx Ultrascale Architecture will be considered. Each Ultrascale CLB contains one slice that can be easily connected to others in order to be combined into more complex functions. There are two types of slices in Ultrascale architecture: logic (SLICEL) and memory (SLICEM). Both contain 8 6-inputs and 2-outputs Look-up-tables (LUT) and 16 storage elements that can be configured as D-type flip-flop or level-sensitive latch. Slices differ in their internal structure and interconnects [13]. Different slices proportions in chips infer feasibility of target application. Basically, LUT element works as a truth table, set of inputs results proper output described in so called LUT-mask. On the other hand storage element preserve values. In reference to SLICEL its resources are mainly utilized in implementation of combinational logic, arithmetic function or signal selection infrastructure (i.e. multiplexer). On the other hand SLICEM enables advance in implementing LUT distributed memories.

Except general purpose logic, designers should leverage other resources. In Ultrascale architectures for on-chip memory resources designers are encouraged to use Block RAM (BRAM) or UltraRAM (URAM) memories except distributed LUT memories [14]. The BRAM in Ultrascale architecture stores up to 36 kb of data. It can be configured as either two independent 18 kb RAMs, or one 36 kb RAM. BRAMs provide variety of configuration features, but as a crucial factor they enable efficient on-chip storage of moderate amount of data and are useful in interfacing between different clock domains, because write interface and read interface can work on different clocking signals. In contrast the single URAM resource can store 288 kb and have dedicated routing resource for cascade connection between multiple blocks. Moreover URAM can cascade inputs and outputs through different clock regions, for BRAM inputs cascade is done with external logic resources and BRAMs cannot be cascaded through different clock regions [6]. Another important resource in Ultrascale Architecture are Digital Signal Processing Blocks (DSP48E2) [15].

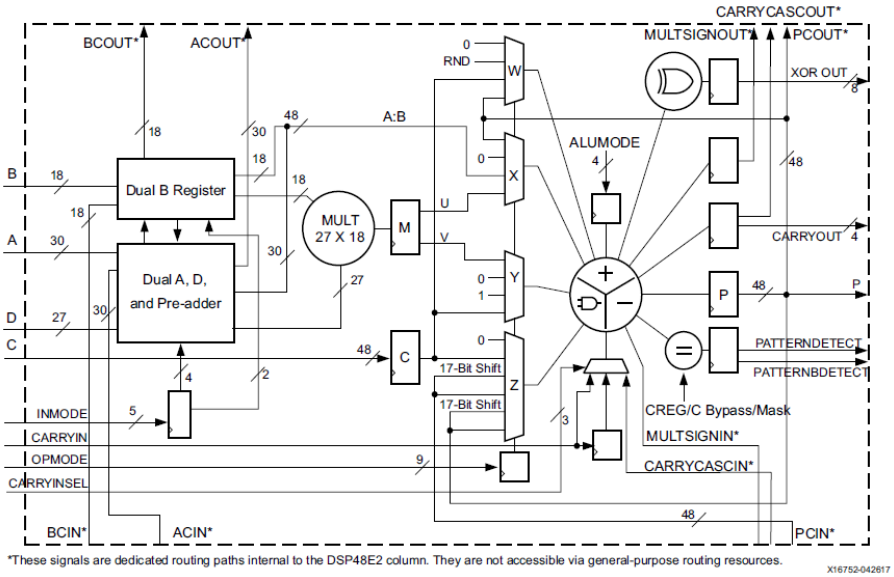

Figure 8 DSP48E2 functional scheme [15]

It supports independent function like:

- Multiply: $P := A * B$
- Multiply accumulate (MACC): $ACC := A * B + ACC$
- Multiply add: $P := A * B + C$
- Four input add: $P := A + B + C + D$
- Bitwise logic functions: $i.e.\ P := A \wedge B$
- Pattern detector and more

Architecture supports cascading multiple DSP48E2 slices for complex arithmetic or Digital Signal Processing (abbr. DSP) without utilizing general purpose logic.

2.2. Design and development flow in heterogeneous reconfigurable systems

Nowadays heterogeneous systems enable dedicated application to use various hardware resources to achieve best performance. Complete system in same silicon chip with computing resources, interfacing resources (like analog to digital converters) and its interconnect is described as System-on-chip (SoC). Unit of logic or electronic design layout is described as Intellectual Property core (IP core), when it is intellectual property of one party. In reference to programmable devices one can diverse components deployed as Soft IP cores or Hard IP cores. Soft IP cores can be implemented with general purpose hardware resources and are fabrication/implementation technology independent. On the other hand Hard IP cores represent hardware components, embedded within FPGA device, mostly used in mixed and analog signal designs (i.e. ADC, SERDES etc.). It enables keeping designs confidential and distributing them as product ready to use with no insights on their internal implementations. Moreover a lot of digital designs IP cores are without restraint on implementation technologies (Soft IP), so with later configuration they can be deployed as IC in semiconductor or configuration in FPGA technology. Basic FPGA design flow can be divided into steps and for Xilinx FPGAs can be presented on following scheme [Figure 9].

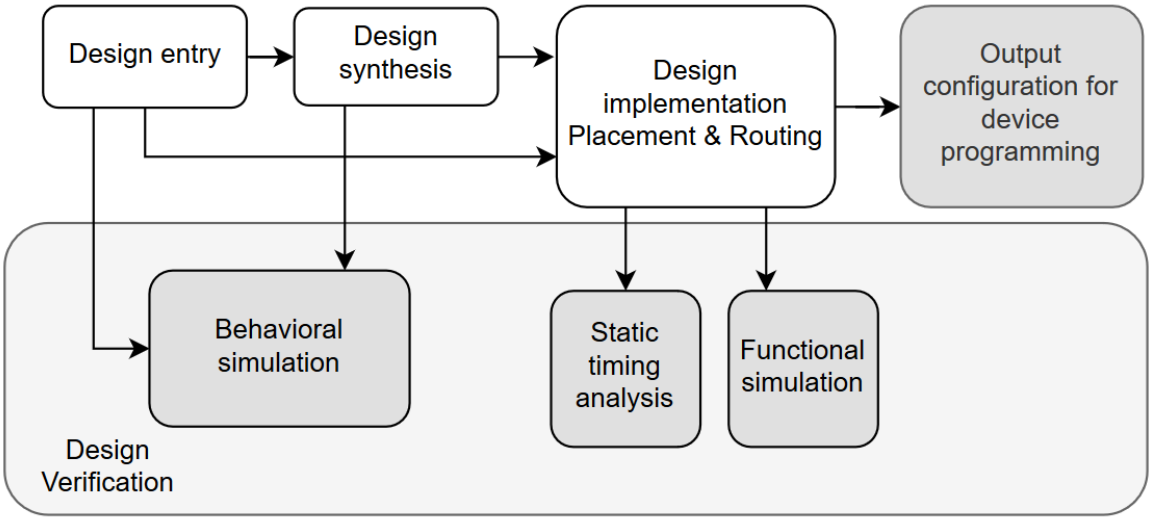

Figure 9 Generic FPGA design flow

In principle FPGA design flows are vendor independent and as a starting point Design Entry states for adding and developing all project source files. Next step is Design Synthesis, where HDL codes and other project sources are checked for syntax and possible violations, then translated to digital electronics description with proper connections made of hardware primitives (usually called netlist). After that, comes the implementation which is about migrating description to physically available on-chip resources. On this stage some optimizations can be done, because resources can be reused in some cases,

moreover some functionalities can be implemented with various available resources (i.e. adder can be realized with LUT or DSP block). At that time also placement and routing of resources is performed. Internal algorithms in FPGA programming toolchain choose viable resources and connections between them in reference to planned strategy and design constraints. Finally design is ready to be saved in proper format for chip programming.

Complementary to design development verification environment is being developed. Aim of verification is to check and warrant bug free designs in their development process and in later deployments. Fundamental rule of verifying components is to check output results, with reference assumptions, as a response to proper stimulus data on its inputs. In behavioural simulation, design is begin verified at the logic level. During implementation, signal propagation and routing delays are calculated basing on generated netlist and selected device. Functional simulation stands for verifying design with delays introduced by the implementation process. Adding delays and timing information to design makes it behave more like real world design deployed on hardware.

In heterogeneous system, except programmable resources there are available other powerful processing units. As an example Xilinx MPSoC contains Application Processing Unit, Real-Time Processing Unit and Graphic Processing Unit except programmable logic and other specialized circuits [16]. In case of designing system architecture it is necessarily important to properly partition functionality to run on programmable logic and fixed silicon structures for best performance and resource utilization.

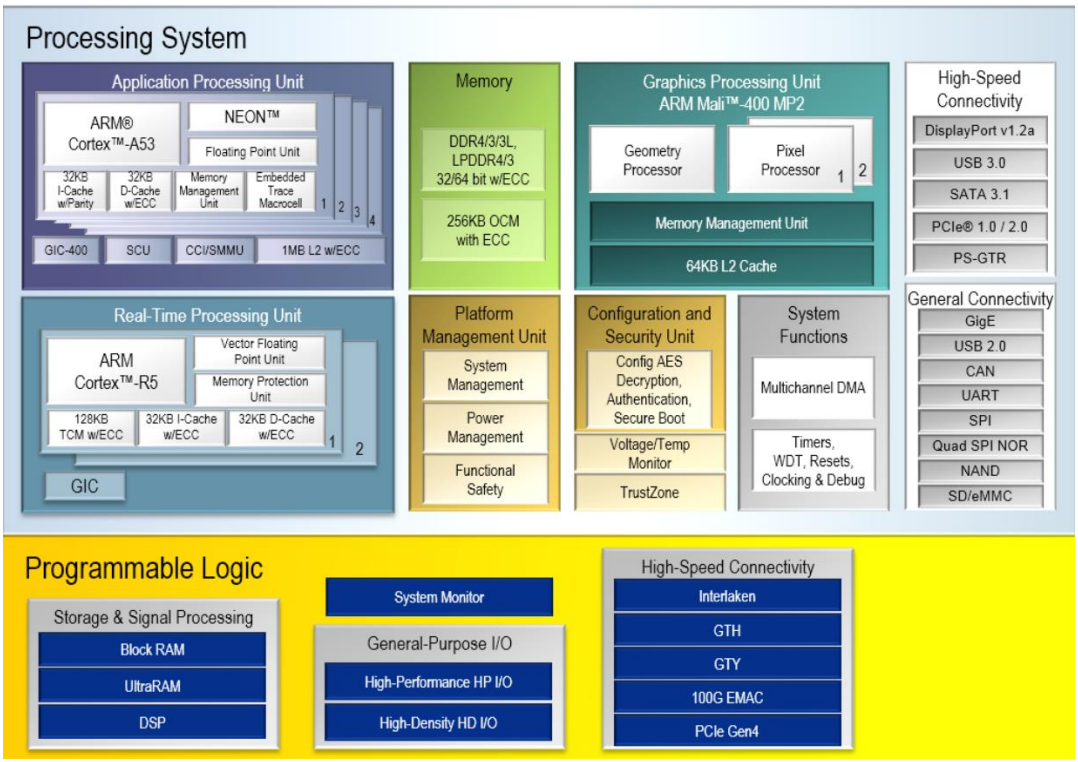

Figure 10 Zynq Ultrascale+ EG device family scheme [17]

As seen on scheme [Figure 10 Zynq Ultrascale+ EG device family scheme heterogeneous system can contain wide range of resources. Considering dedicated application, architect should start with grabbing idea of data rates, calculations complexity and real-time regimes for particular features. Then based on proper assumptions, one can start product development. Software and Hardware project activities can run in parallel but to make it efficient it is worth to design programming model, make assumptions on protocols, build test and verification environment.

Programming model is an abstraction of the underlying computer system that allows for the expression of both algorithms and data structures. It can be implemented with various programming languages, moreover it maps and leverages utilization of hardware resources in accord with requirements.

For designed programming model a software can be developed without hardware up and running, but every change in programming model should be consulted with hardware and software designers. As an example of programming model one can consider Message Passing Interface (MPI), where one or more processes communicate by calling library routines to send and receive messages to other processes.

In reference to protocols, as a set of rules in communication systems, it is considered as a good practice to use standardized ones, for simplified connection with external subsystems and components.

2.3. Verification and testing

FPGAs and SoCs systems grow in resources and become more sophisticated that brings challenges in design processes but also for testing and verification. Moreover, in nowadays FPGA project teams it is quite common to work with Verification Engineers, whose work is it to develop simulation environments for examination of design reliability. Except design complexity, IP reuse drives growth in verification related topics and project activities.

Verification is proving or checking if assumed functions work correctly. In reference to electronics it enables finding bugs and faults in simulation, which iteration is much faster and cheaper than preparation of a physical test/experimental set-up. There is a wide range of possible verification environment development technologies. Basically testbench creation is about applying stimulus for inputs of tested design and checking its output for compliance with assumed behaviour.

As an representative example one can consider verification environment developed with SystemVerilog (SV) and Universal Verification Methodology (UVM) [18]. One of the main advantages of selecting SV is that it combines HDL language with Hardware Verification Language (HVL) in one programming language semantic. Constructs and techniques like classes, inheritance, dynamic objects known in

Object-oriented-programming (OOP) can be used for creating advanced and reusable verification environment. UVM library contains hundreds of classes to advance development of digital and mixed signal verification. It was founded by few semiconductor industry leading companies, they merge and update their own libraries into one bigger and universal.

Fundamental structure of example UVM testbench can be seen on scheme [Figure 11].

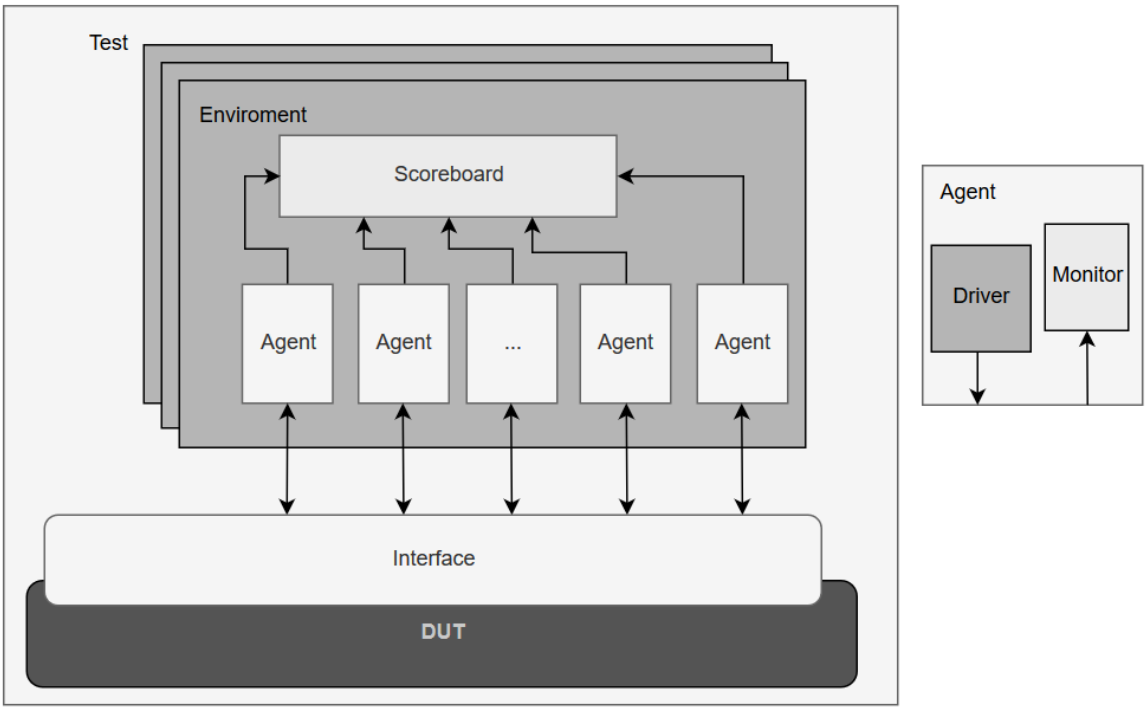

Figure 11 Fundamental structure of Universal Verification Methodology testbench

testcase stands for object of user-defined testcase class, tests can be selected in simulation run command by using command line argument UVM_TESTNAME. Testcase class extends uvm_test class, which is virtual base class for the user-defined tests. In particular test case one can define environments for advanced parameterization of test in different aspects. In reference to environments UVM provides uvm_env virtual class for user-defined environments inheritance. In environments one can instantiate agent connected to scoreboard and Device-Under-Test (DUT) also called Design-Under-Verification (DUV). Their essential work is to drive stimulus via interface to DUT design and monitor outputs passing them to scoreboard for later comparison and analysis.

Essential assumption on UVM simulation execution is phasing [Figure 12].

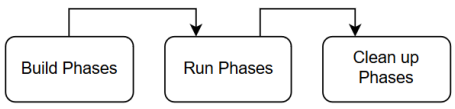

Figure 12 UVM Phasing scheme

Phasing is classified in three categories [18]:

- Build Phases: They are for building and connecting related activities. Objects creation, factory registrations, references passing between objects for later test execution.
- Run Phases: Setting configuration and running simulation. This is the only place for simulation time lined activities like applying stimulus and gathering output data from interfaces.
- Clean up Phases: Clean up and reporting duties for executed simulation are implemented here.

System Verilog provides two possible manners for implementing functionalities methods as: tasks and functions. Main difference between them is that tasks can contain timing control structures unlike functions, which are executed in 'zero time'. Moreover tasks can call other tasks and functions. Phasing is realized with inheritance from uvm classes and overwriting proper phase methods in user-defined classes.

Testing in terms of FPGA and digital circuits design is mostly considered as laboratory testing. This approach is suitable for mature projects or IPs, where common bugs are identified and fixed on simulation of verification environment level. It is because of that, this sort of testing can be costly and time consuming. Some of set-ups can require specialized laboratory equipment. Moreover, single design build for biggest FPGA chips can take a dozen hours. Test scenarios should be quite complex and representative for example usage of application. They are executed on physical hardware platform for compliance check with assumed specification. Adapting Shift-left testing paradigm, which is basically about "testing early and often" it is beneficial to start testing activities early, but with properly simulated designs. One of essential tool for tests of digital electronic, except oscilloscope and signal generator, is Logic Analyzer (LA). This device probes signal and enables later analysis of its archived values. Main disadvantage of that approach is no insights on internal signals, only external pins can by connected to LA. What is more, they are not supporting advanced triggering options for signal samples archiving. FPGA Vendors understand needs for hardware tests and provide IP cores for that purposes. Most common are Integrated Logic Analyzer (ILA) from Xilinx [19] and SignalTap Logic Analyzer from Intel [20]. As a part of IDEs or toolchains for FPGA design that IP cores provide advanced triggering features and insights to internal signals in design. Fact worth considering is fact that ILA and SignalTap utilize on-chip resources. That can influence implementation process and in worst case brings timing issues and shortage of resources.

3. Digital J-PET scanner

3.1. Detector description and nature of signals

Digital J-PET scanner is a whole-body Time of Flight novel tomography developed by J-PET group led by Professor Pawel Moskal at the Jagiellonian University in Kraków. Scanner mechanical structure is designed and manufactured to enable stable and proper fixing of twenty four modules [Figure 14]. Each module contains thirteen scintillation strips arranged one next to another. To each scintillator strip four photomultipliers (PMT) are connected.

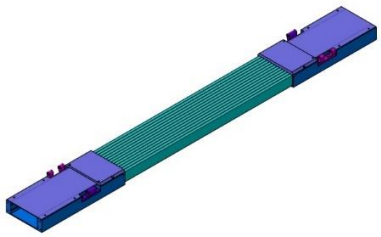

Figure 13 Detecting module of Digital J-PET scanner

Modular design enables custom detector set-ups. Moreover, compact components improve detector mobility. Current ring configuration consist of 24 modules and the inside diameter of detector is around 74 cm. Each module detection area is 46 cm long and 9 cm wide [Figure 13]. Whole detector set-up weigh around 100 kilograms.

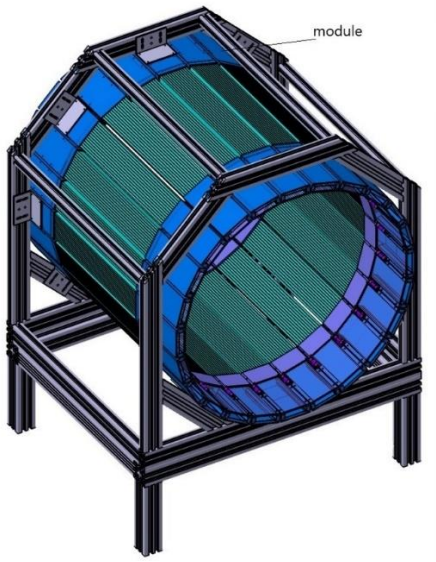

Figure 14 Whole barrel Digital J-PET set-up
source: A. Heczko CAD model

3.2. System architecture

Digital J-PET data acquisition system has hierarchical structure. In bottom-up approach of its characterization first step will be receiving data from FEE readout boards dealing with physical phenomena.

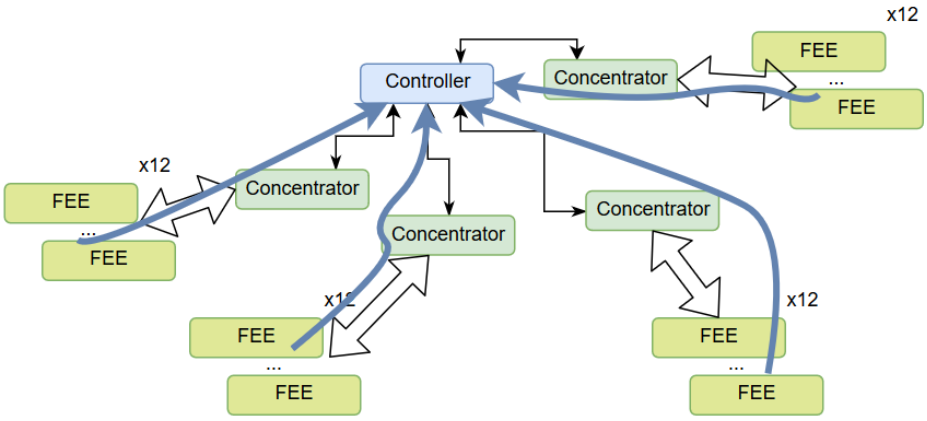

Figure 15 Digital J-PET Conceptual system architecture scheme

Measurement in system goes from FEE modules through Concentrator to the Controller as marked on scheme [Figure 15] with blue arrows. Control communication between Concentrator and its FEEs is bidirectional, white bold arrows indicate that on architecture scheme. Similar, bidirectional communication is between Controller and Concentrators denoted with black arrows. That in result enables Controller to communicate with each FEEs bidirectionally. It is possible to use that infrastructure to communicate components with each other. From a FEE board digital data is sent via optical link to data concentrator board [Figure 15]. Each concentrator board is connected to 6 modules, each having 2 FEEs. In total 12 Front-End boards are connected to each one of 4 Concentrator boards in the system [Figure 15]. That setup enables module-wise processing features to be implemented for monitoring and filtering towards data reduction and later visualization. It is also possible to transmit data to FEE for calibration and control purposes. Each Concentrator board is connected to the Controller board, which gathers data from the entire detector and enables advanced processing (i.e. Real-Time visualization or image reconstruction) and control. From Controller it is possible to communicate with every FEE component in system simply by writing or reading proper memory mapped addresses.

When gamma quanta from annihilation hits scintillator material it may (with possible to estimate probability) emit light which is registered by photomultipliers on opposite ends of a strip. Depending on place where the hit was received in strip arrival time of light differs. Knowing light transmission speed dependencies in scintillator, place of light emission can be obtained. PMT converts received light into

electrical pulse. On each PMT voltage output is distributed to two analog conditioning circuits for later input on LVDS input buffers in Xilinx Artix-7 FPGA chip. LVDS buffer can act like a comparators, when on one of its inputs threshold voltage is applied and on other input signal. Threshold values are determined and applied in calibration procedure, for each PMT output two thresholds are provided.

Concentrator component is implemented on Xilinx Virtex Ultrascale FPGA VCU108 board [Figure 16]. Main advantage of VCU108 is wide range of connectors for multi gigabit transceivers, that allows fast data transmission on many links simultaneously. Moreover Virtex Ultrascale chip contains over 1 million of FF, half of million LUTs and around 60 Mb on-chip memory. That amount of resources enables implementation of complex and parallel processing systems with high-speed communication.

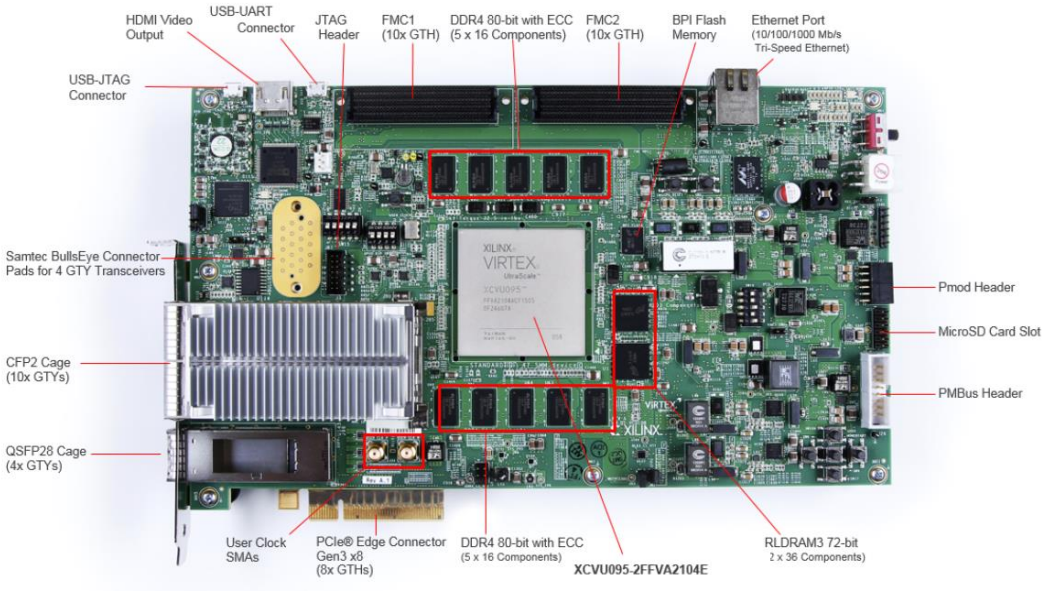

Figure 16 VCU108 board [21]

Controller board functionality is developed on Xilinx Zynq UltraScale+ MPSoC ZCU102 [Figure 17]. It contains heterogenous chip architecture with quad-core Arm® Cortex®-A53, dual-core Cortex-R5F real-time processors and Mali™-400 MP2 graphics processing unit. All combined with programmable logic resources. As distinct from VCU108, ZCU102 contains 2520 DSP blocks and 274080 LUTs. Multi-Processor System enables development of complex software applications for bare metal or operating system layer. That set of resources matches implementation requirements of control, visualization and advanced data processing techniques.

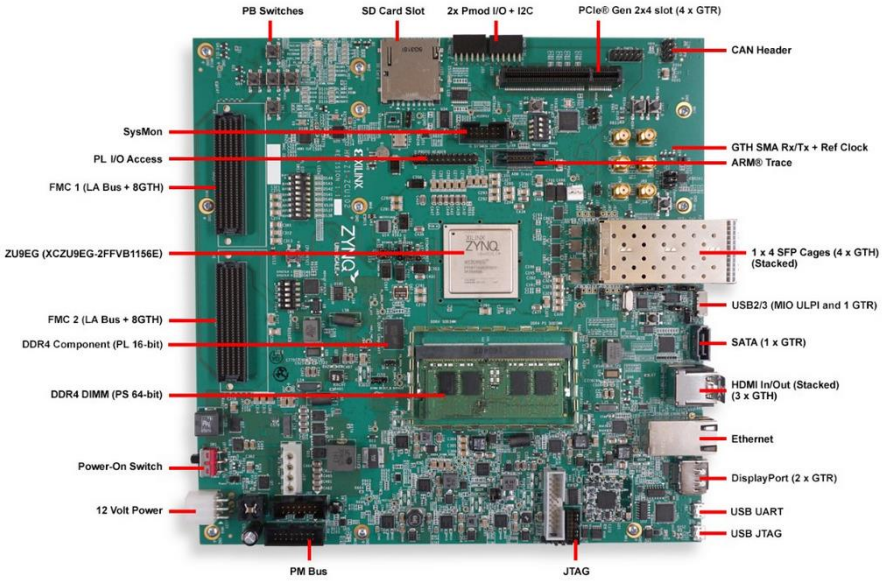

Figure 17 Controller board [22]

3.3. Readout procedure

Digital J-PET uses continuous triggered readout technique to acquire and transmit data from detector to later processing stages. Continuous digitization of analog signals fills buffers, for later transmission when triggered. It is worth to count buffer sizes, radiation intensity and frequency of triggering signal to assure no data loss in case of buffers overflow. Every pulse is described by leading and trailing edge. They are measured with TDC, which output is placed into buffer. Buffered TDC words are in custom binary data format [Figure 18].

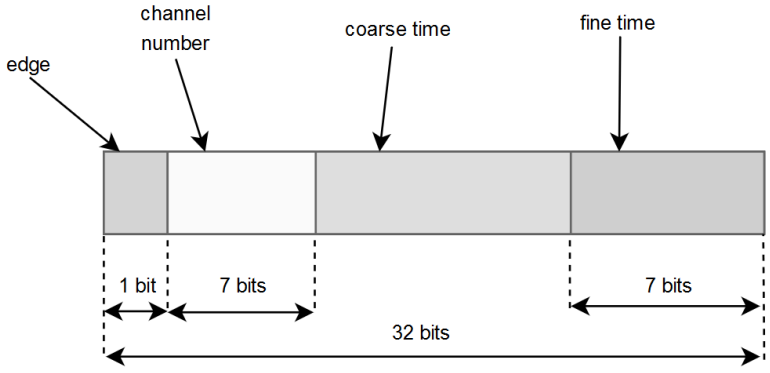

Figure 18 TDC word in Digital J-PET

Edge is matched with single bit value, one stands for rising edge and zero denotes falling edge. Data is measured on two voltage levels called thresholds and from 52 photomultipliers, 4 on each scintillator strip. That results in 104 TDC channels, so channel number value maps each threshold on each photomultiplier on one side of tomographic module. Coarse time and fine time stands for time measurement by counting on 300 MHz clock and TDC-based precise timing information.

Periodic triggering is defined to work with 50 kHz frequency. That means for every timeslot, which in that case takes 20 μs, transmission is occurs. All boards are triggered simultaneously. Transaction header contains 64 bits [Figure 19]. Data width is 32 bits, so first two words stands for board id and amount of TDC word in transaction. Next two words stands for transaction number and error flags.

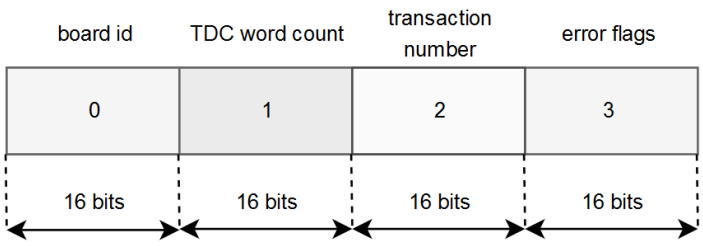

Figure 19 FEE to Concentrator transmission header

Information about board id and amount of data can be used for checking if all data that was sent is properly received. Example usage of event number information and error flags is checking if every transaction that is triggered occurs.

3.4. Data processing stages

First stage in data pipeline is the reception of digitized signals from FEE. Because of using TDC, data consists of words containing timing information. Amount of generated data words depends on calibration parameters and grows with radiation source activity. After transmission from FEE to the Concentrator, data is crossing time domains from transmission clock to processing clock for later histogram building. Basically it is useful to count data from each TDC channel, so the user will be able to see system behaviour before any processing and filtering. As a next step, data from time slot is split into time bins depending on coarse time value for parallel processing [Figure 20].

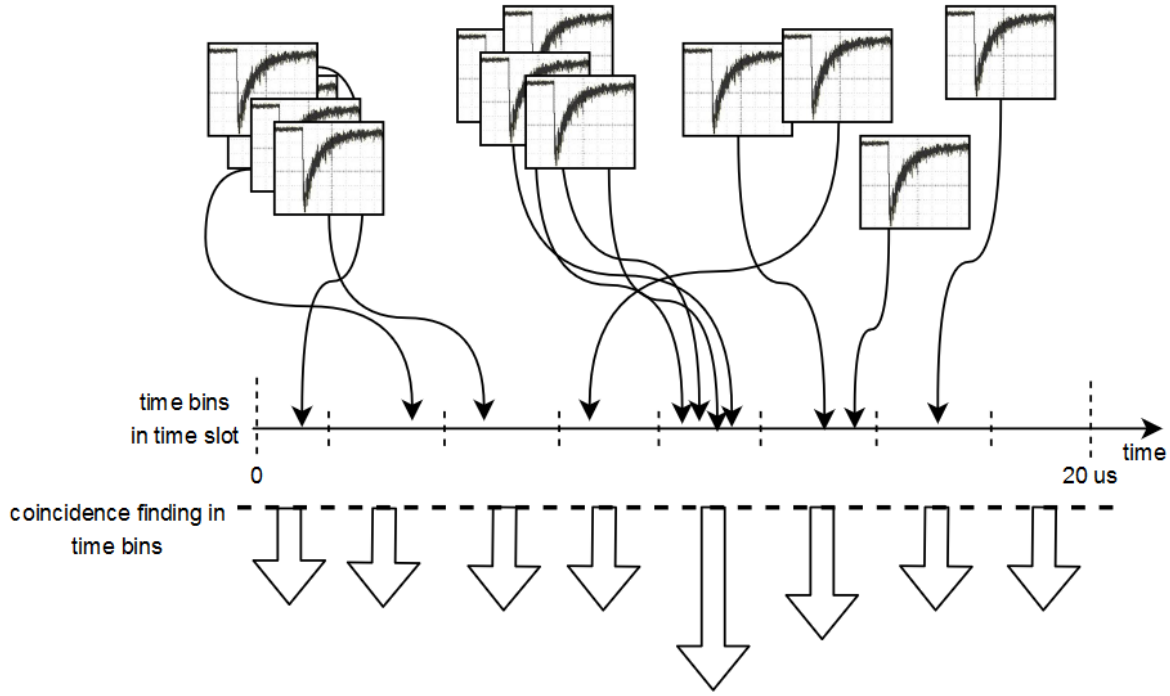

Figure 20 Time bins in time slot split with coincidence finding execution scheme

Time bin size can be parametrized in natural power of 2 values. In example, for 20 μs triggering period coarse time max value on 300 MHz clock is $c_m = 6006$. Assume time bin size of $t_s = 128$ coarse time counter unit. To the first time bin all TDC words with coarse time is in range $\langle 0, 127 \rangle$ will be addressed. In second time bin words with coarse time in range $\langle 128, 255 \rangle$ will be placed and so on. Moreover, that parameters set results in $\lceil c_m / t_s \rceil$ time bins.

For each time bin coincidence finding is performed with later geometrical position in module calculation. Then, the data is gathered to single FIFO module and ready for transmission to the Controller for further advanced processing. Filtered data from the whole detector is duplicated and sent to storage subsystem and a data path that performs LOR extraction modules. On obtained LORs reconstruction algorithms can be executed with hardware acceleration and the output can be visualized directly on display from Controller board.

4. Implementation

4.1. Development environment

Implementation of data pre-processing and image reconstruction kernel is targeted for FPGA technology. Therefore essential tool in development process is Vivado supporting Xilinx programmable devices. It combines all elements of FPGA design flow in Integrated Development Environment (IDE). One of its feature is representing design entry files in form of Block Design [Figure 21].

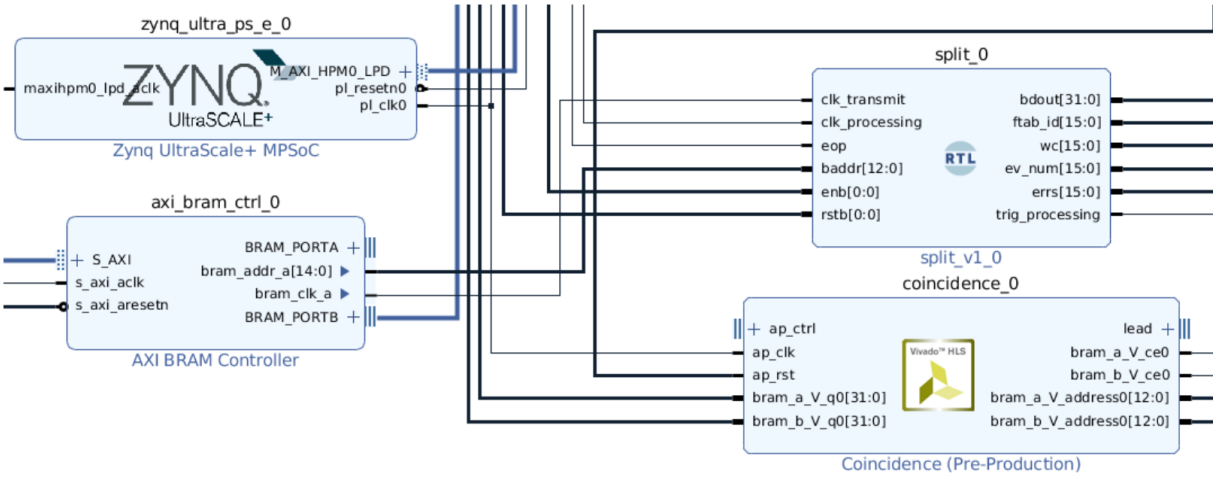

Figure 21 Block design containing HLS, RTL and IP Core components

Block Design enables connection of FPGA design components via graphical user interface (GUI). Moreover, it supports design automation features like:

- Connection Automation: After running connection automation Vivado opens window with available nets and interfaces for connection. Each connection can be customized or connected with default configuration.
- Design Validation: Basic checker on resets, clocks and unconnected pins is executed by running this feature.
- GUI Block parametrization: For wide range of IP cores GUI configuration and parametrization is supported in Vivado from Block Design.

Describing hardware targeting FPGA technology can be performed on few levels of abstraction. In Verilog/SystemVerilog nomenclature the lowest level of circuit modelling is switch level modelling, where designer works with abstractions matching transistors. Modelling on higher level of abstraction is described as gate-level, where digital circuits are built from logic gates. Next modelling level, which

hides logic gate implementation details is called Register-Transfer Level (RTL). RTL models digital circuits in terms of data flow between registers and logical operation. More abstract approach than RTL is Transaction Level Modelling (TLM), where communication details among system components are separated from components implementation details. RTL is common choice in development of synthesizable circuits for FPGA technology, because it combines low-level control on design with moderate abstraction for FPGA specific implementation details.

High Level Synthesis (HLS) is a process of generating digital electronic designs for FPGA technology from common programming language code, mostly C/C++. Digital circuit generation from programming language is controlled with compiler pragma directives and coding styles. In reference to Vivado HLS main aspects of pragma control on generated logic contains [23]:

- Interface synthesis: Defining interface to generated logic component.
- Function inlining: Handling instantiations and reusage of generated logic.
- Kernel optimization: Wide variety of inferring on particular resource utilization, operation chains, latency and resets.
- Pipelining: Control on pipelining and data flow in design.
- Loop unrolling and optimization: Control on dependence between loop iterations, handling iteration execution flow.
- Array optimization: Control arrays storage in memories in reference to enable proper manner of access and better memory utilization.

Except pragmas, generated logic is influenced by coding styles. Usage of proper data types, constant loop bounds, no dynamic arrays are few good practices for HLS projects development. Moreover Vivado HLS contains libraries supporting math operations, linear algebra, video processing, streaming applications and IP cores [23].

4.2. Flow and pre-processing of data for Line-of-response extraction

The first stage of qualifying raw data from detector system as acceptable is to pair output streams from both sides of scintillator strips. When time difference between pulses from both sides on a same scintillator is below certain time boundary, then that pair of pulses is considered as coincidence. Time boundary is determined by the speed of light in scintillator material and its length.

At first, pair of two streams from both sides of scanner module is passed through scalers to count samples per TDC channel in transaction for monitoring purposes.

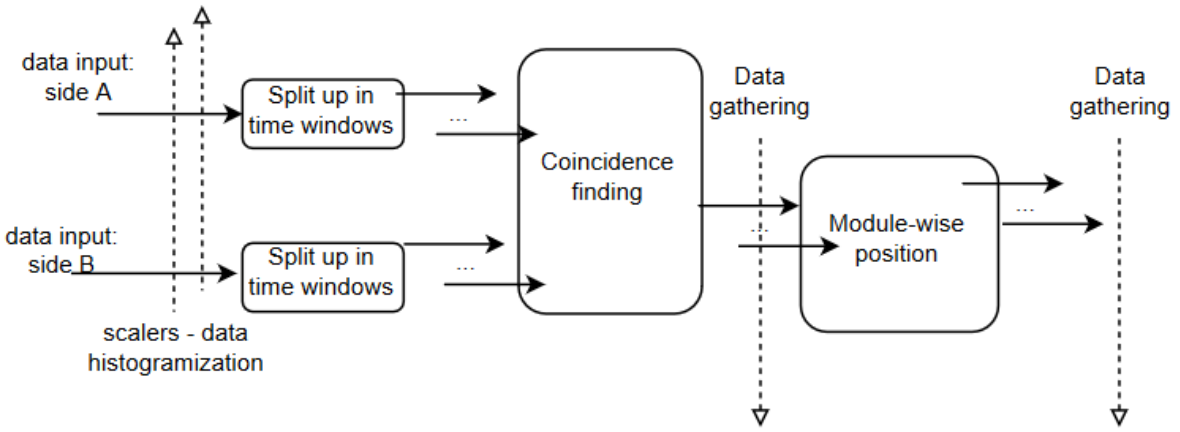

Figure 22 Module-wise data processing architecture scheme

Later data in transaction is split up in time bins with parametrized size covering whole time boundary for a single transaction [Figure 22] (single timeslot). In each time bin coincidence finding is performed in parallel by comparing data from one side with that from the other side of scanner module. In case when a pair has time difference of pulse start below boundary and is from the same scintillator strip, it is passed to output as coincidence. Extracted coincidence pairs are gathered to one FIFO for later transmission to the Controller board. At the same time coincidence results are also passed to calculation of module-wise position. With time difference value, location along scintillator is estimated and based on channel value position on width is determined. Results of positions are also gathered to FIFO module.

Scalers are developed in Vivado HLS, because it is a suitable tool to generate dataflow processing components and registered values will be read by on-chip processor with AXI interface. HLS easily generates proper interface handling on RTL side with histogramization functionality. Split up in time bins is developed in SystemVerilog and it contains a lot of bit-level operations that should be precisely described for best performance and proper timings. Moreover in this module data is transferred to BRAMs with CDC on multibit data bus for later read operations by coincidence extraction stage. Signal “done” of split up operation, which triggers coincidence finders also crosses clock domains. Working on RTL level with non-standard interfacing signalling handshakes is easier to handle and debug than using methods for code generation. Coincidence finding modules are developed in HLS. They read data from BRAM and compare each possible pair combination in time bin. That is essentially a comparison operation to gather coincidences. Module-wise position calculation results in pair of two numbers. First number stands for position on Z axis, which goes along scintillator with its origin on A side of scanner module. Second number stands for position on X axis, it corresponds to scintillator order, but described in distance measure [Figure 23].

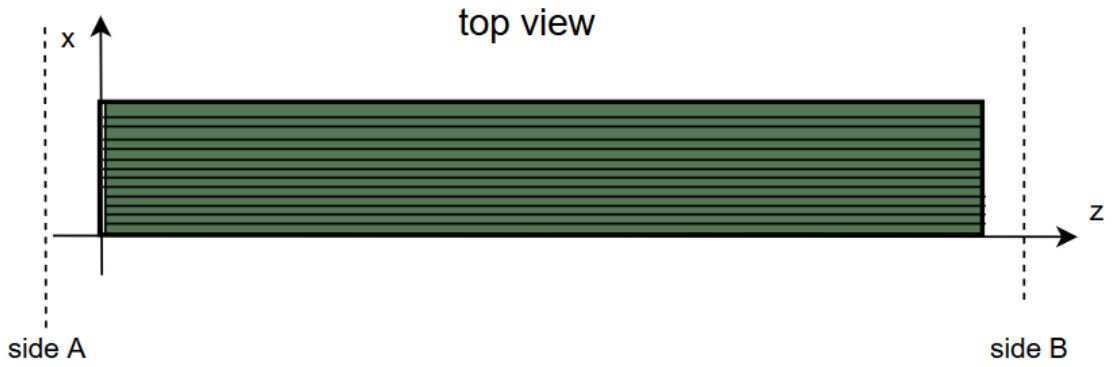

Figure 23 Axis position in reference to module geometry

Module-wise position calculation component basically contains multiplication of scaled time difference with speed of light in scintillator and scanner module geometry mapping for achieving results in centimetres.

4.3. Real-time image reconstruction approach

From a variety of image reconstruction algorithms as an evaluation example of real-time image reconstruction approach 2D Filtered Backprojection method was selected. It is a common, non-iterative method for cross sectional image reconstruction. In reference to hardware implementation, FBP internal calculations are mostly digital signal processing techniques that goes well with FPGA architectures. That premise, next to essential importance of FBP in image reconstruction studies, is the main reason for its hardware acceleration research.

Algorithm implementation input is defined as a list of LORs. As 2 dimensional case is considered LORs are mapped on detector ring plane. In 3 dimensional case smallest element in reconstructed image is called a voxel, in 2 dimensions it is a pixel. Pixel size is inferred by discretization of created sinogram. Bins stands for discretization points, and should be related to the input data and required quality. In general, the more bins, the better resolution. Measurement data from detector is discrete in terms of LORs positions. On detector ring plane LORs can arise only between detecting elements, precisely, scintillator strips. That infers some limitations on bins count preventing oversampling.

From FBP flow accelerated kernel was selected as stages seen on scheme [Figure 24]

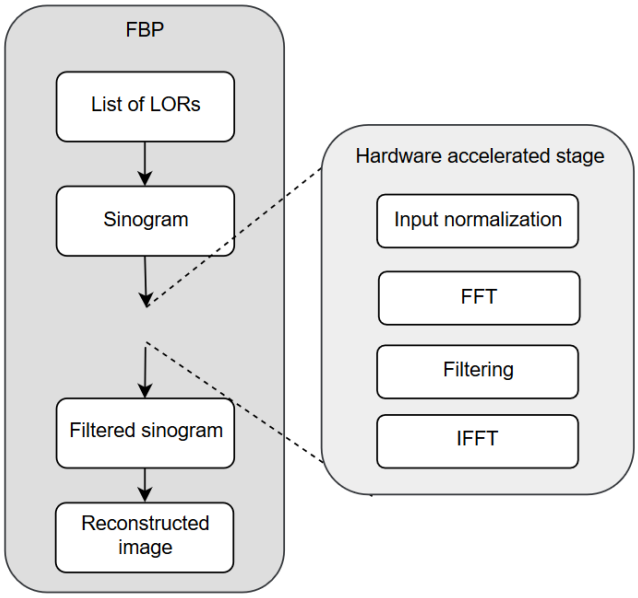

Figure 24 FBP hardware acceleration kernel scheme

FBP reconstruction acceleration kernel takes sinogram data as input. Input normalization is required for later calculation, because input data is bigger than FFT fixed-point data type bounds. Fast Fourier Transform is performed by the FFT IP Core from Xilinx, which works on fixed point data type $Q1.15$. That is 1 bit for signed integer part and 15 bits for fractional part. Every row in sinogram input array corresponds to different rays of fixed angle value in projections. Normalization factor is obtained by summing up all sinogram taking square root of given sum. Later every value in sinogram is divided by the normalization factor. That formula was introduced empirically as a starting point for later data types improvements. It does not result in overflows and preserve large information loss in floating to fixed point conversion. Collaterally sinogram input is buffered to enable pipelined calculations of FFT. Sinogram buffer is declared with synthesis pragma array_reshape that places data in BRAMs with proper split, because from each BRAM only one operation of read or write can be executed. Results of FFT are multiplied with externally passed filter mask. There is no need to calculate filter masks in programmable logic, because it is more suitable to prepare them with proper libraries and frameworks earlier in software and instantiate in logic as a Look-Up-Table. Filtering results are passed to the IFFT core from Xilinx, which works in similar manner to FFT core. Next, element-wise absolute value of filtered sinogram results is taken and data is transmitted to the output buffer to enable IFFT core pipelining.

5. Results

5.1. Data pre-processing

Overall test and analysis environment consists of software execution, running RTL simulation and testing on hardware. Software execution contains reference models and numerical calculations results analysis. Moreover data provided by the software execution of reference models is compared with RTL simulation results and hardware tests products. To correctly run data pre-processing pipeline it is necessarily important to provide inputs relevant to physical process, measurement and transmission method. Input data generation for data pre-processing is structured in three components. First one is the generation of hits locations within detector. Locations are related to position on module plane on X and Z axis. Positions are generated with normal distribution on both axis. This process can be interpreted in terms of real world application as estimation of point source placement on module. Higher number of hits will be registered in position of point source the larger distance from source, smaller amount of hits will be received. Data is generated for both thresholds and transformed into TDC word format. As a result, every geometrical point on module should be represented by two times on both thresholds [Figure 25].

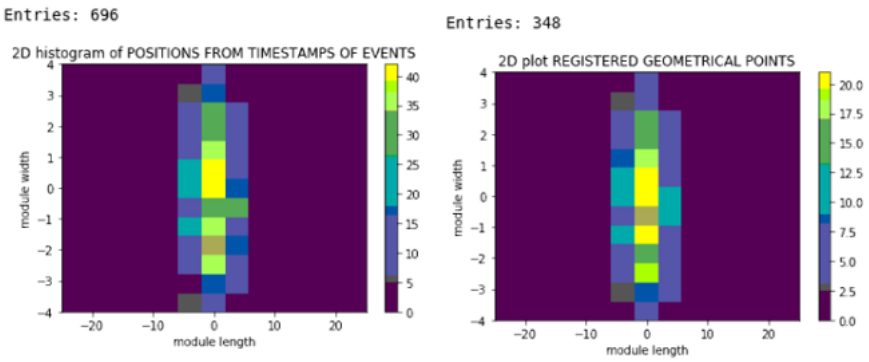

Figure 25 Position on module restored from timestamped data and inital location data

Second component is the acquisition of hits by the detecting system. Positions that match module size boundaries are registered including speed of light in scintillator and two thresholds. Appropriate delays are applied. Later, hits saved in a form of time differences are distributed over timeline. It means that hit occurrence in time slot can be connected with specified time bins [Figure 26].

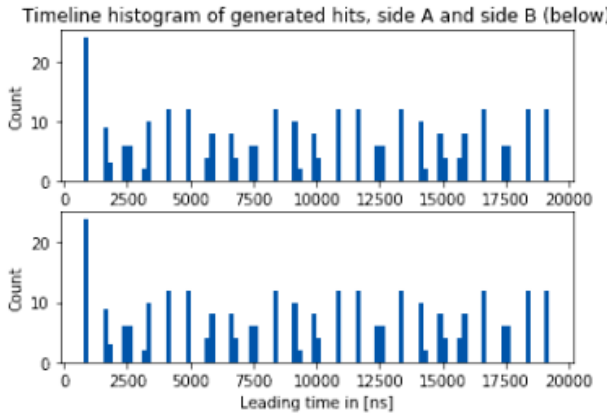

Figure 26 Example time bins distribution with same randomization on both sides

Last component in the input data generator is the formation of transmission packets. Time differences on appropriate TDC channels are encoded in TDC words format. Later noisy data is introduced by removing one element from coincidence pairs. Finally data is shuffled and formed into transmission packet with header. Prepared transmission data is passed to the input of data processing pipeline. Transmitted data split in time bins is passed to HLS functions. First coincidence finding is executed, later module-wise positions are calculated. HLS data processing pipeline outcome is loaded in interactive script for comparison with generated coincidence pairs.

After software model correctness analysis, RTL simulation is executed with same generated input data. Outcome from RTL simulation is checked for compliance with HLS execution results. To guarantee assumed behaviour hardware tests are performed with the same generated input data. To perform hardware tests design was migrated to Zynq SoC chip with processor and FPGA. From the processor data was passed to BRAM, and then transmitted to data pre-processing pipeline. Results from the coincidence finding and module-wise calculations are gathered in FIFOs and read back by the processor for comparison with the reference one. When failure occurs proper message with invalid data is send to display on host computer via Universal Asynchronous Receiver-Transmitter (UART).

5.2. Image reconstruction

Reconstruction of image in PET tomography is based on LORs. From coincidences in terms of hit receival on a single detector, Line-Of-Responses can be obtained by considering the entire detecting setup and time coincidences between reconstructed module-wise hit positions. Image reconstruction can be separated in two parts. First one is the construction of a sinogram in an pipelined, event by event manner, by processing LORs one after the other. Second part is the execution of the computing kernel. When sinogram is filled with satisfactory amount of LORs, image reconstruction can be launched [Figure 27].

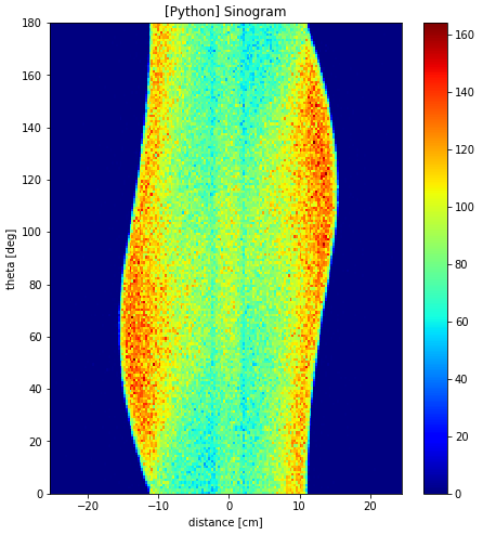

Figure 27 Example sinogram

FFT is performed on sinogram, resulting with a plot containing frequency information. Different scales are present because FFT IP Core input data type in range from -1 to 1 [Figure 28].

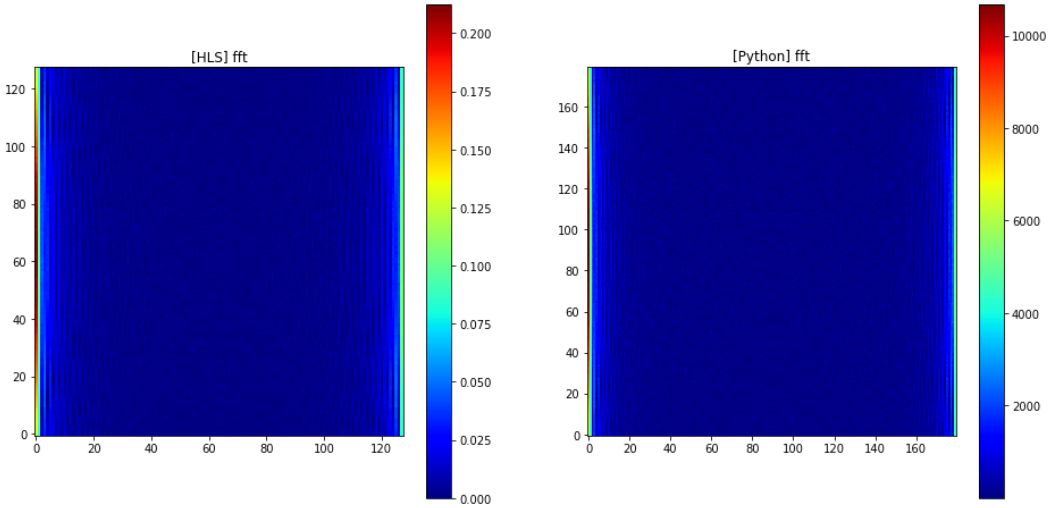

Figure 28 FFT of sinogram from HLS and Python reference model

Later filtering is performed by multiplication in frequency domain. As an example, RamLak filter is selected with cut-off frequency of 0.9 .

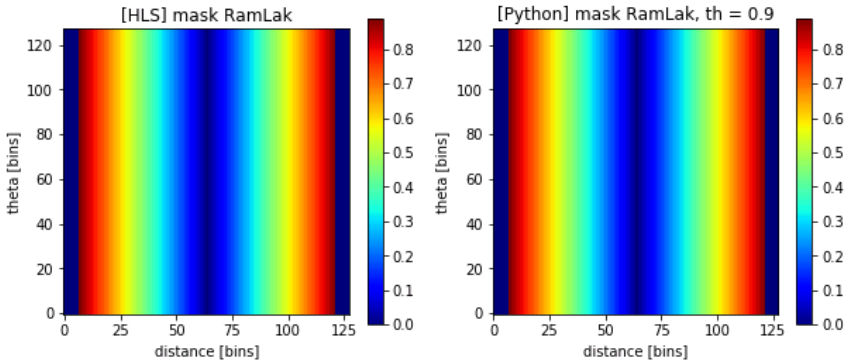

Figure 29 RamLak filter mask for cut-off frequency= 0.9

Filtering with cut-off frequency of 0.9 acts like a band pass filter, where low frequencies are attenuated and highest removed [Figure 30]. Presented HLS example FFT module is configured to operate on normalized values in range from -1 to 1, because it uses 16 bit fixed-point data type with 1 integer bit and 15 fractional bits. That results in not bit exact results with different scales on plots.

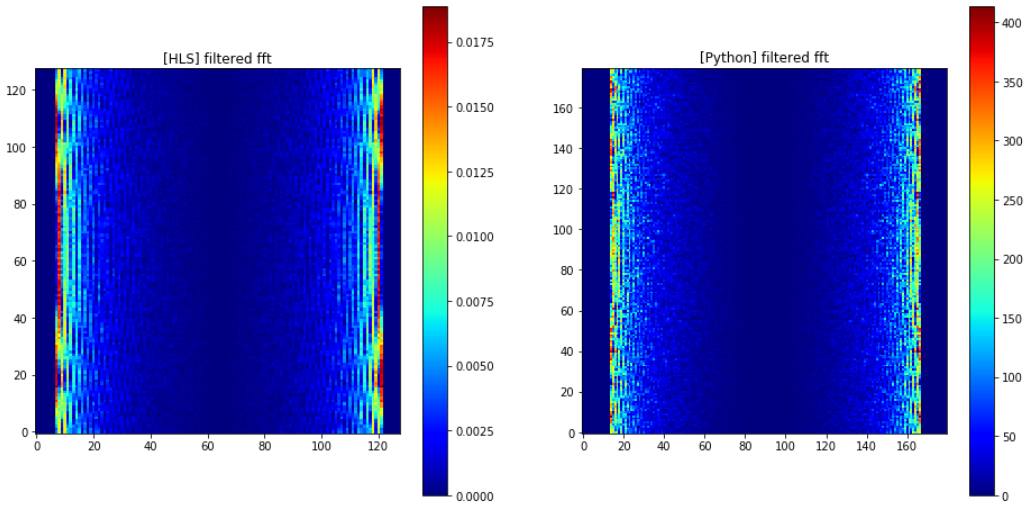

Figure 30 Sinograms in frequency domain after filtering

Later IFFT is performed, which results in a recovered sinogram [Figure 31]. After filtering, moderate and high frequencies are amplified.

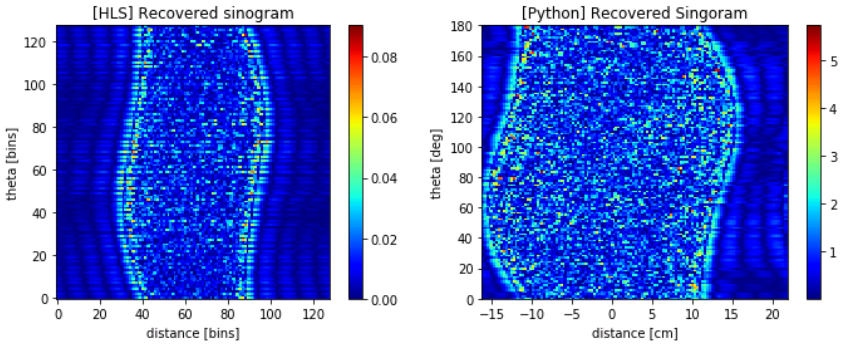

Figure 31 Filtered sinograms from FBP method

Parallelization of iterative calculations is essentially realized by unrolling loops. If there are no dependencies between iterations of loop it is possible to run them in parallel. Unroll factor corresponds to the number of iterations that are executed in parallel. It is presented on [Figure 32], where one can recognize 4 instances of fft_top function, all executed at the same moment. By increasing unroll factor latency is decreased and resource utilization is assumed to grow.

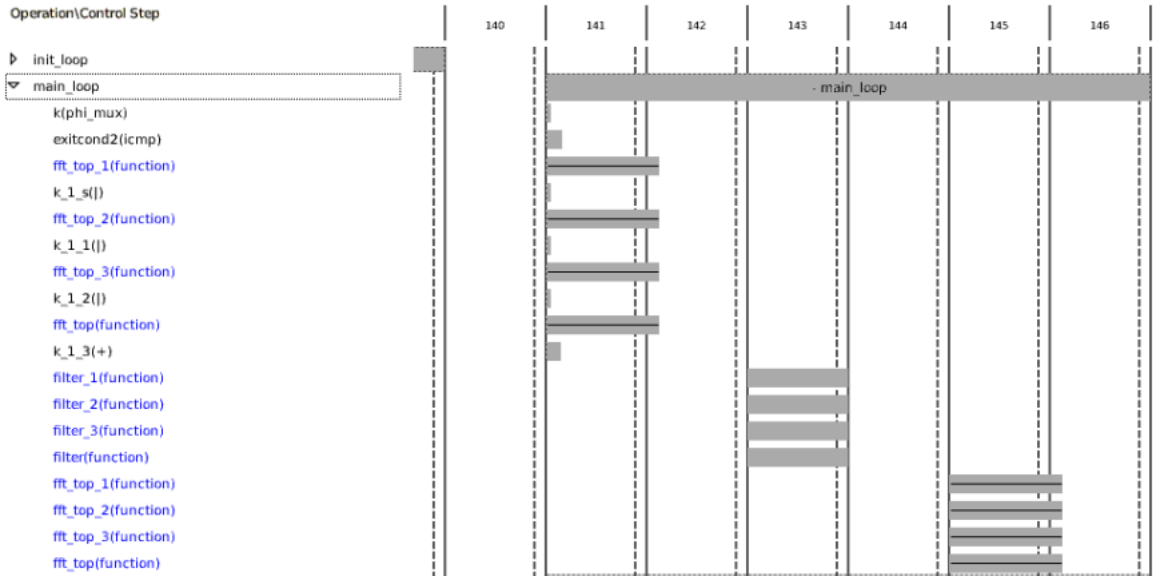

Figure 32 Loop unroll by factor of 4 for fft_top, filter and fft_top consecutive operations.

Calculation kernel can be configured for fixed-point or floating-point data type implementation. For main types of operations used in kernel like addition and multiplication fixed-point hardware implementation requires less resources what derives faster clocking. Main disadvantage of fixed-point data type in terms of calculation kernel is small range of possible values. That corresponds to the need for data normalization to prevent overflows. Floating-point data type can store data from higher range

of values, but numerical operations like additions and multiplications are in general more resource consuming.

Analysis of kernel parallelization for fixed-point configuration is presented for 1, 2, 4, 8 unroll factors with pre-processing normalization [Table 1]. Presented values consist of sequential processes for reading input sinogram and writing generated sinogram to the output and parallelized computations in between. The sequential manner of input and output is due to the type of the interface, which allows to read or write single bin per clock cycle and this dominates the overall latency and interval measurements. That is why increasing unroll factor does not result in proportional reduction of latency.

Unroll factor	Latency $[ms]$	Interval $[ms]$
1	4.07	4.07
2	3.59	3.59
4	3.35	3.35
8	3.26	3.26

Table 1 Latency and interval in fixed point implementation

Normalization is based on dividing each element by maximum value in row, that corresponds to different distances for fixed theta. That normalization provides values in range from -1 to 1 suitable for configuration using fixed-point data type. Latency is the value that depends on conditional statement in normalization. Moreover normalization in this case consumes resources and additional clock cycles. It also introduces dependency, because data from input must be firstly normalized for later calculations.

That is why unroll can be easily performed only from stage after normalization. Flow of data in that manner of normalizing does not allow to introduce pipelining.

On the other hand floating-point configuration does not require normalization. Input data can be buffered and passed to later unrolled loop of kernel calculations. Without normalization the kernel can fully profit from the parallelization of its calculations, without being biased by sequential computations.

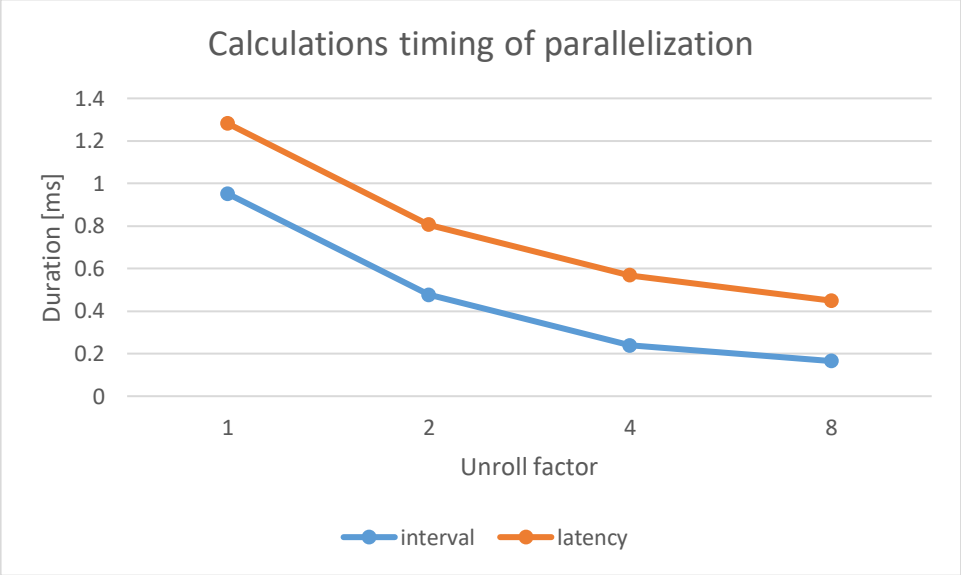

Figure 33 Floating point implementation calculations timing

Moreover that manner of flow enables pipelining using dataflow compiler pragma. Overall performance can benefit from dataflow because the calculations can occur while input and output data is being transmitted. On example of unroll factor 4 configuration latencies and interval durations are as follows:

Before dataflow pragma optimization		After dataflow pragma optimization	
Latency $[ms]$	Interval $[ms]$	Latency $[ms]$	Interval $[ms]$
0.65	0.65	0.57	0.24

Table 2 Data flow pragma optimization comparison

With dataflow optimization first results will be delivered after 0.57 [ms] and after that, every kernel execution in pipelined manner will provide results every 0.24 [ms]. In terms of reconstruction 0.24 [ms] corresponds to 4166.6 Hz refresh rate. These results indicate time margin for non-kernel data processing to reconstruct image in pipeline with low-latencies.

Resource utilization in reference to increasing unroll factor has a growing trend. BRAM Memory resources are slowly growing. Fixed-point configuration utilize around 78% of them, for floating-point

it is 71.5% of total BRAM memory on-chip resources. Flip-flops tend to scale linearly and for unroll factor of 8 FF utilization is around 10%. Most crucial resources in kernel are LUTs, because FFT cores are configured to utilize them for calculations.

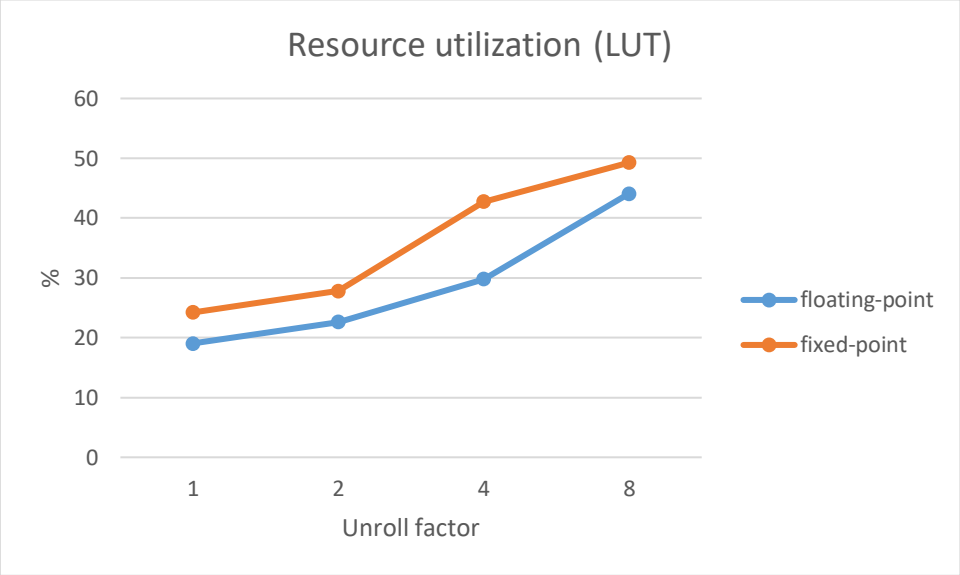

Figure 34 Resource utilization (LUT) plot

Prepared kernel for both configurations and considered unroll factor values will fit into Controller Board chip resources. Except presented resources only few DSP blocks (out of 2520 available on chip) are currently utilized kernel.

6. Conclusions

6.1. Development considerations and problems occurred

In modern FPGA work flow designers are encouraged to use various techniques from software development and electronic design automation background to deliver complex digital components faster and in a more flexible way. This trend is supported by new tools, libraries and frameworks for software applications to better match the hardware or even generate hardware description based on application source code. Vivado HLS is an example of using software development techniques in hardware design process. Functions targeted for hardware acceleration can be executed in the same way as software functions, which enables regular programming test and debug features. For generated logic tests, Vivado HLS provides co-simulation to confirm that RTL matches software behaviour. C/C++ code that executes synthesised HLS function is translated to RTL testbench and run in RTL simulation manner.

6.2. Future plans

Next steps of project development activities can be classified in two categories. First is connected with improving data processing for LOR extraction and FBP optimizations. Second category is related to research on other methods of image reconstruction.

To extract LORs from paired coincidence on both sides of scintillator there is a need for appropriate detector geometry mapping with registered hit channel information from TDC words. Based on that, coincidence in terms of positron decay can be obtained regarding hit receival on two detectors in short time period. That will result in a pair of two points representing a LOR, with additional information on Time of Flight. LORs should update the sinogram one by one for later image reconstruction. In FBP implementation FFT and IFFT IP cores can be configured in few other ways. They can work with different data types and utilize resources in different manner. That can possibly result in elimination of input normalization and enabling greater flexibility in resource utilization management.

Regarding other methods of image reconstruction, it is worth to consider Time of Flight FBP (TOF FBP) [24]. It provides better reconstruction quality than classical FBP. Moreover, in TOF FBP sinogram filtering, which hardware implementation is presented in this work, is extensively used. Other interesting approach to image reconstruction methods is using Kernel Density Estimation (KDE) [25]. KDE based methods can omit filtering by directly working in projections domain. Main idea in these methods is to enhance ROR information with kernel operator and accumulate outcome. That in result will reconstruct image.

6.3. Summary

The aim of this work, which was to explore hardware acceleration techniques for tomographic data processing and visualization, is achieved. Data processing was developed and tested for coincidence finding and module-wise position determination. In reference to visualization component of this work, FBP method was evaluated for hardware acceleration.

Implementation details are presented in 4. Chapter, where design of module-wise data processing and image reconstruction approach is described. Delivered data processing components provide insight to data pipeline on three stages: counting data after receival, finding coincidences and hit positions calculations within module. FBP sinogram filtering kernel determines feasible way of algorithm acceleration prototyping.

Nowadays image reconstruction techniques in PET tomography are in general accelerated with general purpose processing units. Hardware implementation of reconstruction and data pipeline can possibly enable wide range of new application, where real-time regime, compact design and energy efficiency are crucial factors. Whole project including results from 5. Chapter confirms suitability of using FPGA technology for data processing and image reconstruction acceleration.

Bibliography

[1] J. F. Y. Santiago, Positron Emission Tomography with Computed Tomography (PET/CT), Springer, 2015.

[2] T. Buzug, Computed Tomography From Photon Statistics to Modern Cone-Beam CT, Springer, 2008.

[3] A. Girach and R. C. Sergot, Optical Coherence Tomography, Springer, 2016.

[4] M. K. Miller and R. G. Forbes, Atom-Probe Tomography: The Local Electrode Atom Probe, Springer, 2014.

[5] M. Reiser, W. Semmler and H. Hricak, Magnetic Resonance Tomography, Springer, 2008.

[6] J. Harvey and e. al., Techniques and concepts of High Energy Physics V, Plenum Press, 1990.

[7] M. Zvolsky, *Lectures on Tomographic Image Reconstruction.*

[8] P. A. Toft, The Radon Transform - Theory and Implementation, 1996.

[9] D. H. Garces, W. T. Rhodes and N. Peña, "Projection-slice theorem: a compact notation," *Journal of the Optical Society of America,* May 2011.

[10] L. Seung-Wan, L. Chang-Lae, C. Hyo-Min and e. al., "Effects of Reconstruction Parameters on Image Noise and Spatial Resolution," *Journal of the Korean Physical Society,* pp. 2825-2832, October 2011.

[11] H. K. Tashima and K. H. e. a. T., "Restoration of lost frequency in OpenPET imaging: comparison between the method of convex projections and the maximum likelihood expectation maximization method," *Radiological Physics and Technology,* pp. 329-339, July 2014.

[12] J.-P. Deschamps, G. D. Sutter and E. Cantó, Guide to FPGA Implementation of Arithmetic Functions, Springer, 2012.

[13] Xilinx, UltraScale Architecture Configurable Logic Block User Guide - UG574 (v. 1.5), 2017.

[14] Xilinx, UltraScale Architecture Memory Resources User Guide - UG573 (v. 1.10), 2019.

[15] Xilinx, UltraScale Architecture DSP Slice User Guide - UG579 (v. 1.8), 2019.

[16] Xilinx, UltraScale Architecture and Product Data Sheet: Overview - DS890 (v. 3.10), 2019.

[17] Xilinx, "https://www.xilinx.com/content/dam/xilinx/imgs/products/zynq/zynq-eg-block.PNG," [Online].

[18] Accellera Systems Initiative (Accellera), Universal Verification Methodology (UVM) 1.2 User's Guide, 2015.

[19] Xilinx, Integrated Logic Analyzer v6.2 LogiCORE IP Product Guide - PG172, 2016.

[20] Intel, Intel Quartus Prime Pro Edition User Guide, 2019.

[21] Xilinx, "https://www.xilinx.com/products/boards-and-kits/ek-u1-vcu108-g.html#hardware," [Online].

[22] Xilinx, "https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html#hardware," [Online].

[23] Xilinx, Vivado Design Suite User Guide High-Level Synthesis - UG902 (v2017.1), 2017.

[24] M. Conti, B. Bendriem and a. et, "First experimental results of time-of-flight reconstruction on an LSO PET scanner," *Physics in Medicine and Biology,* pp. 4507-4528, 2004.

[25] R. Y. Shopa, "Application of Kernel Density Estimation for Image Reconstruction in J-PET Scanners of High TOF Resolution," *IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC),* pp. 1-3, 2018.

[26] W. R. Leo, Techniques for Nuclear and Particle Physics Experiments A How-to Approach, Springer-Verlag, 1994.