SUMMER STUDENT PROGRAMME 2025

รายงานการเข้าร่วมโครงการนักศึกษาภาคฤดูร้อนเซิร์น
ระหว่างวันที่ 8 มิถุนายน - 31 สิงหาคม 2568
ณ เซิร์น กรุงเจนีวา สมาพันธรัฐสวิส

นายธนวินท์ ทีวะเวช

วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

Preface

This report has been prepared to summarize the outcome of my internship project, DFX4ML: Enable Dynamic Function eXchange for Machine Learning, at the European Organization for Nuclear Research (CERN). This project covers CERN internship activities, my technical insight into my project, my preparation before the internship, and my daily notes during the internship.

This report is intended to be a digital footprint of my internship journey. On the technical side, it documents the reasoning on why we apply any specific hardware design technique and demonstrates problem-solving on both hardware/software issues. We also provide in-depth hardware architecture design to enable the successor designer to understand and be capable of reproducing my hardware implementation in the future. On the personal side, I aim to demonstrate how I manage to solve project issues or challenges for each project task. I realize that problem-solving skills are the most important for a hardware developer and the hardest part. I would be glad if my report helps anyone solve problems faster.

Finally, I hope this report will benefit everyone who is interested in hardware chips and system design, design automation, and Machine Learning. Any mistakes or inaccuracies in this report are solely my responsibility, and I sincerely apologize for them.

Tanawin Devaveja

Acknowledgement

I would like to express my deepest gratitude to the European Organization for Nuclear Research (CERN) and the Thailand-CERN Collaboration Project under the Royal Initiative of Her Royal Highness Princess Maha Chakri Sirindhorn for granting me the opportunity to participate in the CERN Summer Student 2025.

I am profoundly grateful to my supervisor, Mr. Nicolò Ghielmetti, for his continuous support and guidance throughout my research project during the internship. The original idea of this project came from him. I would also like to sincerely thank Mr. Sioni Summers for his helpful advice and insightful feedback throughout the project.

Furthermore, I would like to express my sincere gratitude to Assistant Professor Norraphat Srimanobhas and Ms. Tichapat Upatham for their guidance and support in preparing me for living abroad and assisting me in addressing various challenges during the internship process.

It would be remiss of me not to acknowledge Assistant Professor Krerk Piromsopa for his encouragement and for providing a letter of recommendation on my behalf.

Finally, I would like to express my heartfelt appreciation to my family for their love and support.

Disclaimer

This manuscript was refined using Grammarly and ChatGPT, solely for grammar and clarity checking. All ideas and expressions are entirely the author's own.

References were generated using EndNote under a license provided by Chulalongkorn University.

Table of Contents

Preface ..... A

Acknowledgement ..... B

Disclaimer C

Table of Contents ..... D

CERN Internship Overview 1

Summer Student Activity 1

Site Visit 1

Poster and Oral Presentation Session 2

My Project at CERN 3

My Weekly Presentation 3

DFX4ML: Enable Dynamic Function eXchange for Machine Learning 5

Abstract 6

Introduction 6

Background 7

1. HLS4ML [2] 7

2. Dynamic Function eXchange 8

Methodology 9

1.Machine Learning Chopping 10

2. The Magic Architecture 11

3. HLS4ML system design automation 16

Testing 17

1. Resource Usage 18

2. Execution Performance 19

3. Reconfigure Performance 20

Current Progress 21

Conclusion 24

References 25

Preparation Diary Report 26

Diary Report 29

Biography 67

References 69

CERN Internship Overview

Every year, CERN offers opportunities for bachelor's and master's degree students in physics, mathematics, and engineering to join research teams at CERN in Geneva, Switzerland, for a period of 8 to 12 weeks. Before the internship begins, students receive a project brief outlining the work they will undertake at CERN. In addition, the assigned project is typically matched to the student's field of study and personal research interests. In addition, a student is typically assigned to a CERN research experiment group such as CMS, ATLAS, ALICE, and LHCb.

Summer Student Activity

Not only the assigned project, CERN offers opportunities for other activities related to their research, such as site visits, open-lab, lectures, poster sessions, and oral presentations.

Site Visit

CERN offers summer students the opportunity to experience four prestigious working sites: the ATLAS experiment, the Synchrocyclotron, the Antiproton Decelerator, and the Data Center. In the current year, students were able to visit both the ATLAS control room and the Synchrocyclotron Museum (Figure 1). However, we were required to choose between visiting the Antiproton Decelerator or the Data Center.

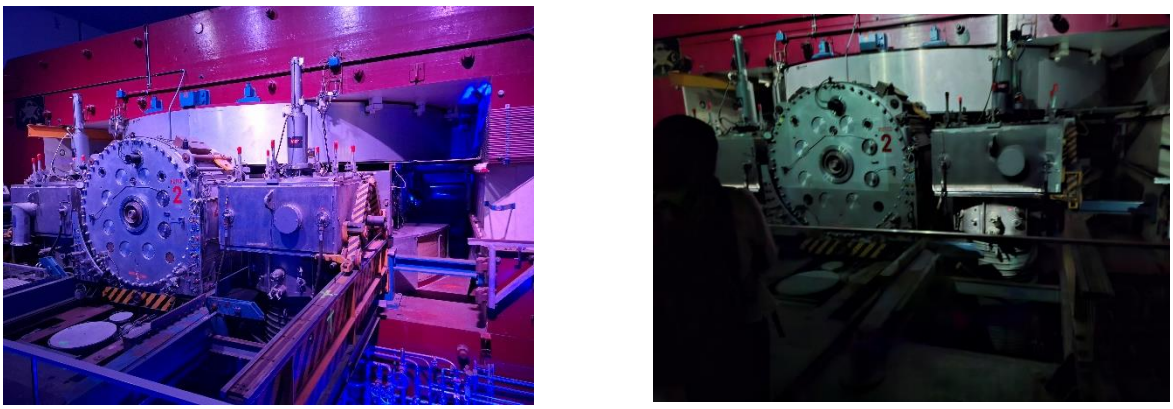


Figure 1 Synchrocyclotron site visit

“The 600-MeV Synchrocyclotron (SC), which came into operation in 1957, was CERN’s first accelerator. It provided beams for CERN’s first experiments in particle and nuclear physics.” [1]

Figure 2 CERN data center (left), Atlas Higgs Boson Counter Display (right)

The figure on the left shows CERN’s data center, which is used to store data collected from experiments and also serves as a computing engine for research. The figure on the right shows the ATLAS HIG BOSON counter from the ATLAS experiment.

Poster and Oral Presentation Session

CERN offers summer students the opportunity to share their work with fellow students or anyone interested. For both sessions, the presenter is limited to 30 seats for each section. The poster and oral presentation sessions are established on 24 July 2025 and 7-9 August 2025, respectively.

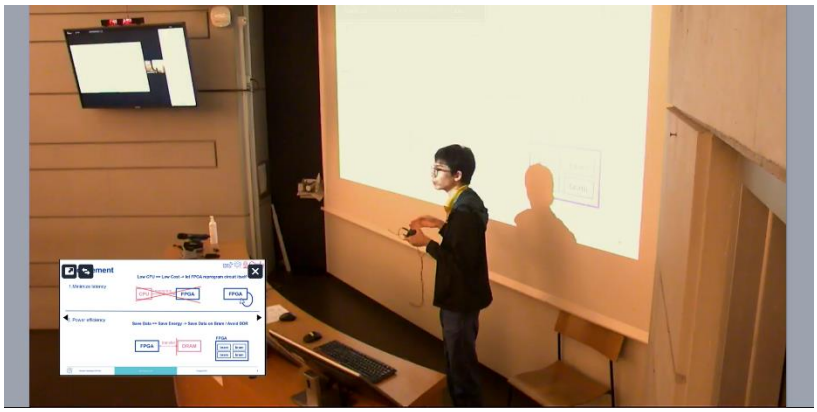


Figure 3 My Oral Presentation Session

My Project at CERN

Topic: Enable Dynamic Function eXchange for Machine Learning

During the internship, the research project did not relate to particle physics experiments. The project mainly focuses on hardware chip design for a Field Programmable Gate Array. However, our technique may benefit physics research because it was applied to HLS4ML. As far as I know, HLS4ML plays an important role in LHC trigger and data acquisition systems.

Specifically, my work focuses on exploring the partial reconfiguration technique for HLS4ML. Partial reconfiguration technique is the method to separate the full, large neural network that does not fit the FPGA into multiple smaller pieces of a fit neural network. The main challenge is how we can manage the execution sequence efficiently.

In a working environment, this work is mainly supervised by Mr. Nicolò Ghielmetti. I update my working progress, receive a great comment, and get task guidance every Tuesday at 10 a.m. The weekly presentation slide was shown in the link below:

My Weekly Presentation

1. introduction to dfx (17 June 25)
    a. https://docs.google.com/presentation/d/1Am3aTRLVb8oUEHwq6NsfDWjCSIcmV9Vhl9EVlZogGCY/edit?usp=sharing
2. hls4ml tutorial (24 June 25)
    a. https://docs.google.com/presentation/d/1ov-sdazNeJuJ-ITOpo67QsL3cMXTcI0XyDrqmq9msYE/edit?usp=sharing
3. magic sequencer initialization (30 June 25)
    a. https://docs.google.com/presentation/d/1k8fed_mw5JrLjl2BhR2NmtYwjlIHxvhC3IUr_6QCMLg/edit?usp=sharing
4. DFX4ML chop the HLS into multiple parts (7 July 25)
    a. https://docs.google.com/presentation/d/17XdL4yeKDuxsatxQx2rltuI0HCJt72h_n87rxaG68lc/edit?usp=sharing
5. skip connection (15 July 25)
    a. https://docs.google.com/presentation/d/1ncXNhVfGJSJwoBSOCet1cMqj4AcvyYFR94TzUkK_TIo/edit?usp=sharing
6. magic streamer (22 July 25)
    a. https://docs.google.com/presentation/d/1T-h_4ERSRtcp-obVTphe4Y7mLSzVBprWum1-6GDel1o/edit?usp=sharing

7. magic streamer (27 July 25)
    a. https://docs.google.com/presentation/d/1F5tK0TS-VKrrMX7-Purq7FiEsPUIuqp991WBt-E7Ko0/edit?usp=sharing
8. unet deploy (5 Aug 25)
    a. https://docs.google.com/presentation/d/14w3Ydswqq6FwoOLzY5VtpzuIXF4GnGBao8IP59XkkBw/edit?usp=sharing
9. summer student session (8 Aug 25)
    a. https://docs.google.com/presentation/d/1HL--JlnzKIcSLdkYqvUN4uLGV1plE5BW1nBsDv4yX4o/edit?usp=sharing
10. reuse factor problem (12 Aug 25)
    a. https://docs.google.com/presentation/d/1HL--JlnzKIcSLdkYqvUN4uLGV1plE5BW1nBsDv4yX4o/edit?usp=sharing
11. new backend implementation (19 Aug 25)
    a. https://docs.google.com/presentation/d/1UPkIo5vhluPU7FHXPBKI3blJbR4ugBKjgis2pzBsloI/edit?usp=sharing
12. HLS4ML group presentation (22 Aug 25)
    a. https://docs.google.com/presentation/d/1gRxVzgEY_64keN2yhuQTR3etD4acOYJCc8EIpzW18ic/edit?usp=sharing
13. Vitis Unified partial (26 Aug 25)
    a. https://docs.google.com/presentation/d/1WDid5pljWQmJtzJpoflimXPm5rOZFGd8gybl2tvF95c/edit?usp=sharing

DFX4ML: Enable Dynamic Function eXchange for Machine Learning

By: Tanawin Devaveja

(Chulalongkorn University)

Supervised by:

Supervisor : Nicolò Ghielmetti (CERN)

Co-Supervisor : Sioni Summers (CERN)

The European Organization for Nuclear Research (CERN)

CERN Summer Student Program 2025

Abstract

The machine learning workload demand in edge computing is increasing over time. Typically, edge computing resources such as power, computing power, memory, and price are limited due to its design and environment. In contrast, machine learning workloads require an enormous computing workload because their mathematical operations, such as depth-wise convolution and max pooling, require a lot of computing power. Therefore, a Field Programmable Gate Array (FPGA) can be utilized to optimize these mathematical operations. FPGA can customize both mathematical operations and utilize dataflow from layer to layer. However, for a large machine learning workload, an FPGA cannot provide the hardware resources for the workload requirement.

This project aims to solve this problem. We apply the partial reconfiguration technique that dynamically reconfigures the FPGA to reuse the hardware resources while executing the machine learning workload.

Introduction

Since edge computing has become more popular for deploying machine learning workloads, the Field Programmable Gate Array (FPGA), a reprogrammable integrated circuit, is usually used to execute the machine learning workload. FPGA is useful for these workloads because FPGA can customize the dataflow of the neural network layer on the chip; moreover, its mathematical operations, such as convolution and max-pooling, can be optimized to reduce resource usage and execution time.

Although an FPGA benefits machine learning workload for edge computing, deploying a large machine learning workload is sometimes not possible because the large workload requires too many hardware resources. Even though there are resource optimization techniques such as quantization, pruning, and configuring the reuse factor, overutilizing resources still limits the workload deployment.

In this project, we aim to apply the partial reconfiguration technique for a large machine learning workload. Partial reconfiguration is the procedure to reconfigure a portion of the FPGA fabric. In addition, Xilinx calls this procedure Dynamic Function eXchange (DFX) [1]. If we can reconfigure the multiple tiny workloads into the same resource, we can reuse the resource to

optimize resource usage. To apply this technique for a machine learning workload, we convert the full neural network graph into multiple smaller graphs. In addition, this conversion is done using the HLS4ML framework. To manage this system, we provide the MAGIC architecture to make the FPGA schedule the machine learning, reconfigure the FPGA, control the intermediate layer data flow, and store intermediate data.

As a result, in deploying a simple dense workload, the resource usage report shows a significant reduction in LUT, in BRAM, and in registers. More details in the testing section.

Background

1. HLS4ML [2]

It is a framework that converts various machine learning model vendors, such as Keras [4], TensorFlow [3], PyTorch, and QONNX, into high-level synthesis projects from vendors like Vivado [6], Vitis [7], and One API. High-level synthesis is the methodology for synthesizing a hardware chip at the register transfer level by providing algorithmic code in a high-level programming language, such as C/C++.

As depicted in Figure 4The model graph is an intermediate representation of the converted model. It is stored as a graph that contains multiple nodes linked together. Each node represents the neural network layer. Each node comprises a layer attribute and a connection between layers.

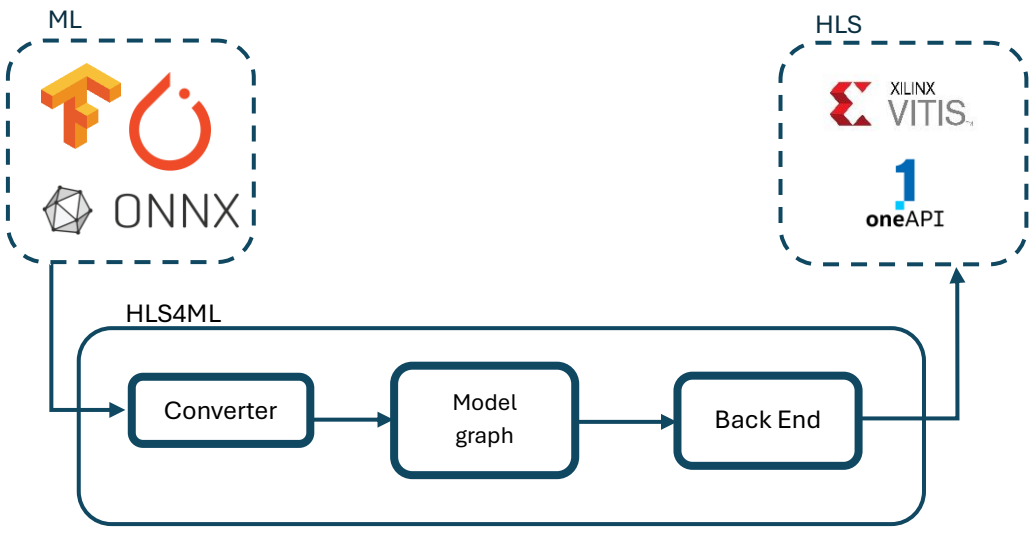


Figure 4 HLS4ML Framework Workflow

For instance, a 2D convolution node typically comprises the following elements: kernel size, variable precision, input matrix size, kernel value, resource usage strategy, input/output port name, reuse factor, and base generator (for backend generation purposes). This is essential data for the backend. This is the necessary metadata for the backend workflow.

For the Back End, the graph from the previous stage will be gathered using the mentioned metadata to generate the HLS project corresponding to the vendor specification. The workload does not involve only graph conversion, but also generating the subsystem related to the hardware infrastructure and the pre-production system.

In practice, to have basic support generation, the backend should generate these four main subsystems.

1. HLS Neural model: the hardware synthesizable model. Typically, it is written in C/C++.
2. Bridge simulator: The simulator allowed system-level simulation of the model graph using the generated HLS model
3. Co-simulator: the simulator allowed register transfer level simulation using a post-synthesis HLS model
4. FIFO - depth optimization: an optimization technique for reducing the FIFO between layers.

2. Dynamic Function eXchange

A Field Programmable Gate Array (FPGA) is a reprogrammable logic circuit. The mechanism used to reprogram is by loading the FPGA's configuration memory with a bitstream file. Typically, a bitstream file is composed of compiled instructions that relate to each resource.

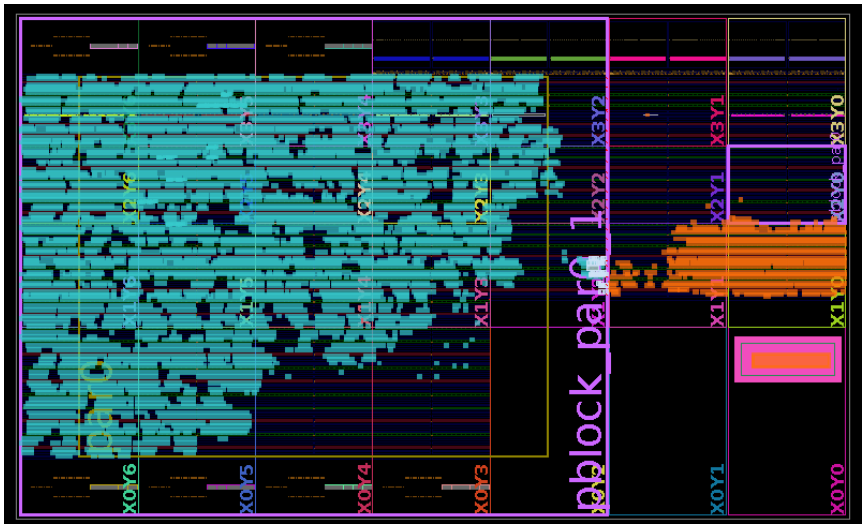


Figure 5 ZCU102 FPGA fabric's cells visualization

For Xilinx FPGA, the instruction can customize a partial region of the integrated fabric. For instance, at the Figure 5In the dynamic region, the blue cells are reconfigured at different times from the outside region. On the other hand, the static region, the orange cells, can operate normally while the dynamic region is reconfiguring.

Methodology

To exploit the dynamic configuration feature for Machine Learning workload, we categorized the working process into three following three minor steps.

1. Machine Learning model chopping: It is supposed to convert the HLS4ML intermediate graph into a corrected minor workload.
2. Magic architecture implementation: This system is built to manage multiple chopped graphs.
3. HLS4ML system design automation: This integrates the three steps above into the HLS4ML framework to make the entire process automatic.

1.Machine Learning Chopping

The Model graph in Figure 6 Contains a single whole graph built for a single deployment on an FPGA. After using the multigraph feature in HLS4ML, it will be chopped at the red line into multiple graphs.

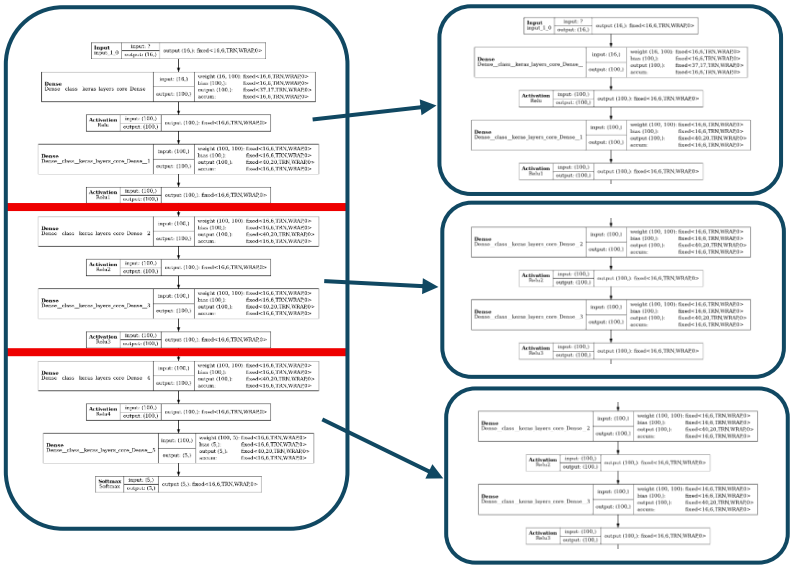


Figure 6 Neural Network Separation Visualization

After chopping the graph down, it raises some questions. That is, who will schedule the graphs to deploy in the FPGA, when the graph is identified as a finished state, especially for multiple outputs, where the output from each subgraph will be stored. This highlights the necessity of adopting a magic architecture

2. The Magic Architecture

After having multiple subgraphs from HLS4ML, the subgraphs have to be controlled in the correct sequence. The Figure 7 illustrates the mechanism while the model is performing inference.

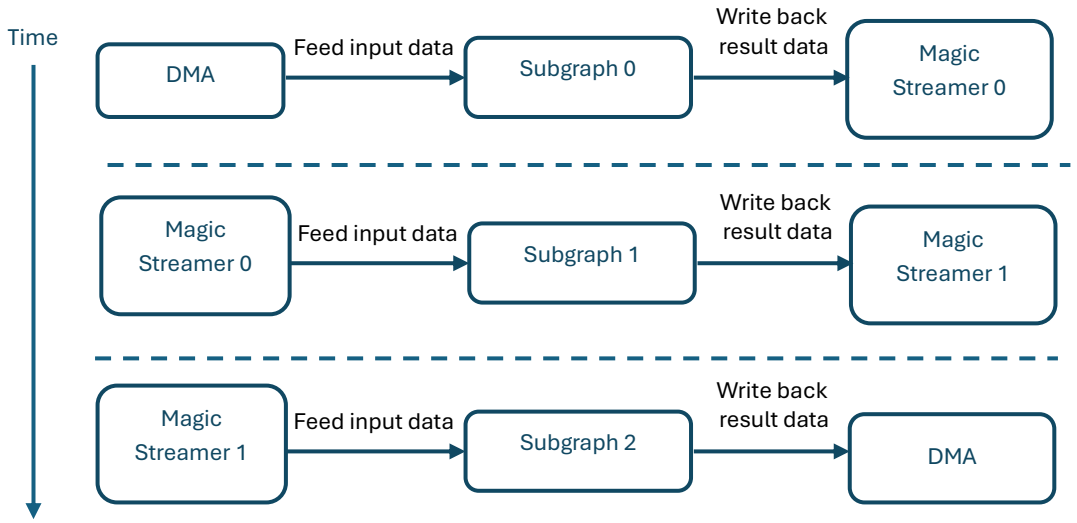


Figure 7 Machine Learning Inference Procedure on Magic Architecture

Each subgraph requires its own dedicated data storage to store temporary data while the next subgraph is reconfiguring in the FPGA. We called the storage Magic streamer, and we have the Magic Architecture to control these sequences. Figure 8 illustrates the overall architecture implemented in the FPGA fabric.

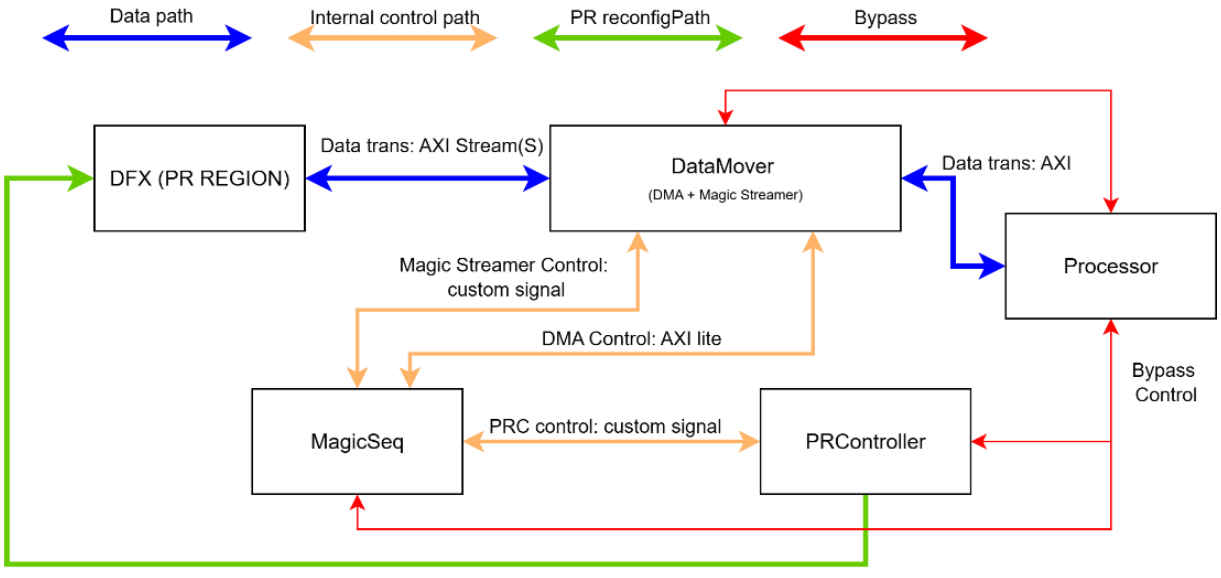


Figure 8 Magic Architecture Diagram

DFX (PR Region) is a reserved partition on the FPGA fabric on which the HLS4ML subgraph will be dynamically implemented. For instance, the pink rectangle labeled pblock_par0_1 in Figure 5 It is a partial reconfigure region in this case. DataMover is composed of Magic streamers and a DMA block. The DMA block is used to transfer the data between the main memory and the PR Region. In Contrast, Magic Streamers is used to temporarily store the data from the PR Region and push the data back to the next module that will be reconfigured at the next sequence. PRController is the DFX controller IP provided by Xilinx. The controller will retrieve the bitstream file that was pre-allocated in DRAM and write it to the Internal Configuration Access Port (ICAP) port for PR Region reconfiguring. In addition, ICAP is the dedicated port located in the FPGA chip. It is used to write the FPGA configuration memory. The memory will directly affect the FPGA's logic cells. MagicSeq (Magic Sequencer) controls PRController on when DFX should be reconfigured and which Reconfigure module should be reconfigured. It also controls and tracks the data stream flow from the DMA and Magic Streamer.

2.1 Magic Sequencer

Magic sequencer is the IP located in the static region, outside of the DFX. The IP has the dedicated Register architecture depicted in Figure 9 to control and track the status of the PRController and DataMover (Magic streamers + DMA). The register architecture is composed of 2 main register banks. Bank 0 is used to contain the status of the system, and Bank 1 is used to contain specific data for each reconfigurable module.

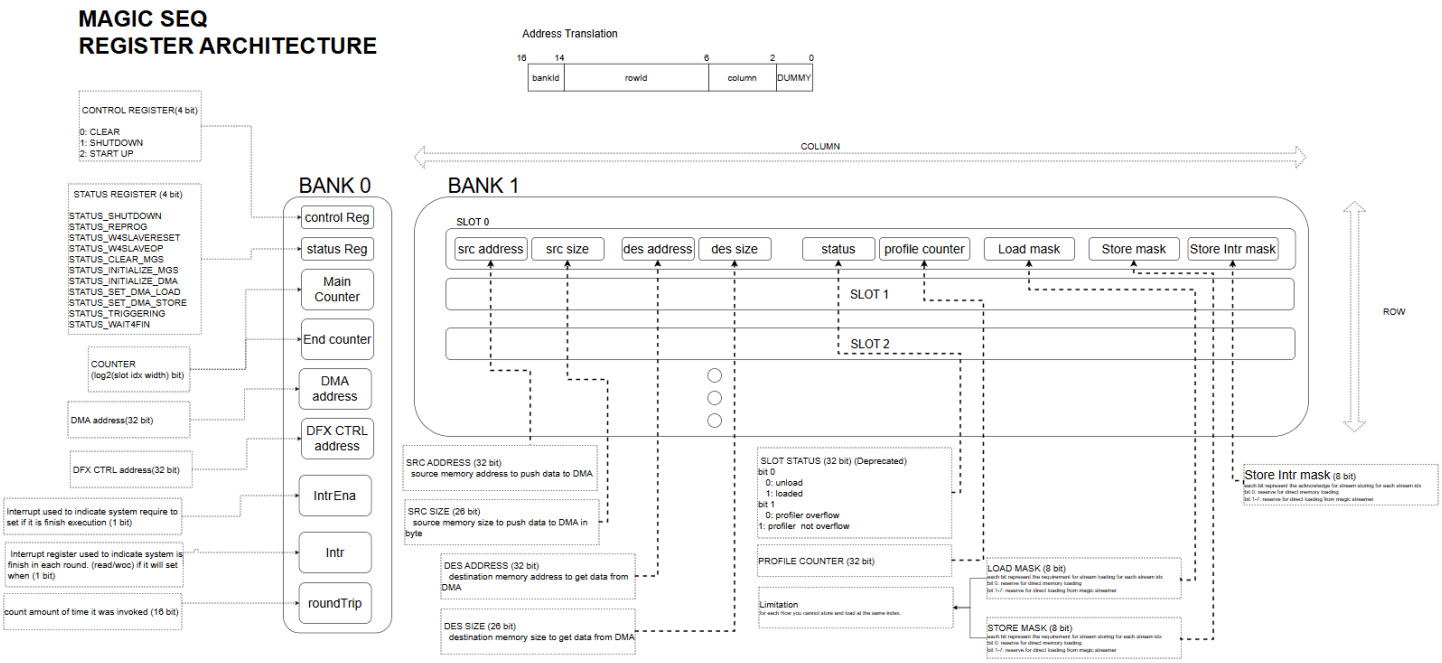


Figure 9 Magic Sequencer Register Architecture

Bank 0 is composed of the following registers:

1. Control Register (0x0000): It is a write-only register used to enable the processor to send a control command.
2. Status Register (0x0020): It is a read-only register used to enable the processor to read the current status of the magic sequencer and track the current executing procedure of the system.
3. Main Counter (0x0040): It is a read-only register used to track the current executing reconfigure module index of the system.
4. End Counter (0x0060): It is a read/write register used to specify the last index of the reconfigurable module in each execution batch.
5. DMA Address (0x0080): It is a read/write register used to store the address of the Direct Memory Access IP.
6. DFX CTRL Address (0x00A0): It is a read/write register used to store the address of the DFX IP. Actually, this address is deprecated.
7. INTR Ena (0x00C0): It is a read/write register used to store the setting that requires the system to interrupt the processor when the system finishes executing each batch of reconfigurable modules.
8. INTR (0x00E0): It is a read/write (write-on-clear) register used to store the interrupt status of the system when the system finishes executing each batch of reconfigurable modules. It will not set if the INTR Ena register is not set first.

9. Round Trip (0x0110) It is the read/write register used to specify recurrent times to infer the model.

Bank 1 is composed of several slots and each slot provides the necessary data for each reconfigurable module to execute. Each slot is composed of the following registers.

1. Src Address (BaseSlot + 0x00): It is 32 32-bit read/write register used to store the address of the inference data in main memory. If the inference data is from Magic streamer, this register can be set to any value.
2. Src Size (BaseSlot + 0x04): It is a 32 32-bit read/write register used to store the size of the inference data in main memory in Byte units. (Caution: the size must not exceed the setting in the DMA IP)
3. Des Address (BaseSlot + 0x08): it is 32 32-bit read/write register used to store the address of the inference result in main memory. If the inference result is from Magic streamer, this register can be set to any value.
4. Des Size (BaseSlot + 0x0C): It is a 32 32-bit read/write register used to store the size of the inference result in main memory in Byte units. (Caution: the size must not exceed the setting in the DMA IP)
5. Status (BaseSlot + 0x10): It is deprecated
6. Profile Counter (BaseSlot + 0x14): It is 32 32-bit read/write register used to track the reconfiguring time of the current module.
7. Load Mask (BaseSlot + 0x18): It is a one-hot encoding read/write register. Each bit indicates the index of the Magic streamer to push into the reconfigurable module, except bit 0. Bit 0 is used for DMA IP.
8. Store Mask (BaseSlot + 0x1C): It is a one-hot encoding read/write register. Each bit indicates the index of the Magic streamer to retrieve the intermediate inference result from the reconfigurable module, except bit 0. Bit 0 is used for DMA IP writing back into main memory.
9. Store Intr Mask (BaseSlot + 0x20): It is a one-hot encoding read/write register. Each bit tracks the status of the magic streamer that finishes storing the intermediate inference result. Except bit 0 is used for DMA IP write-back status.

All registers in bank1 can be accessed only when the Magic Sequencer is in the shutdown state.

The Magic streamer has the following procedure to reconfigure the module and execute the inference data.

1. When reset is set, the system is in the SHUTDOWN state
2. When the system receives the start command, the system enters the REPROG state. This state triggers the DFX IP to start reconfiguring the partial reconfigure region
3. After that, the system waits for the reconfiguration until it finishes in W4SLAVERESET and W4SLAVEOP state.
4. Next, the system enters the CLEAR_MGS state to clear the temporary metadata in all magic streamers
5. Next, the system enters INITIALIZE_DMA to reset the interrupt signal in the DMA IP
6. If the current slot requires the DMA to load inference data from main memory, the system will enter the SET_DMA_LOAD state
7. If the current slot requires the DMA to store the inference result to the main memory, the system will enter the SET_DMA_STORE state
8. After DMA configuration is finished, the system will enter the TRIGGERING state. This state is reserved for further pre-execution.
9. The system will then wait for all inference results to finish at the WAIT4FIN state. After it has finished, the system will enter the shutdown state.

2.2 Magic Streamer

The magic streamer is used to store the intermediate result of each subgraph. The Bram, the on-chip memory block, is used to store the data. We use the Bram instead of transferring the data to main memory via DMA because the on-chip memory transfer uses less power compared to the main memory transfer. Figure 10 illustrates the Magic Streamer architecture

The streamer is composed of 3 meta to track the status of data transfer between Bram and the reconfigurable module via Stream Receiver and Stream Transceiver based on AXI stream protocol:

1. Amount Store register: track the size of the intermediate inference result that is streamed into the BRAM.
2. Amount Load register: track size of intermediate inference result that is streaming into the reconfigurable module.
3. Interrupt Holder register: It is a single-bit register indicating that the system has finished storing the data stream.

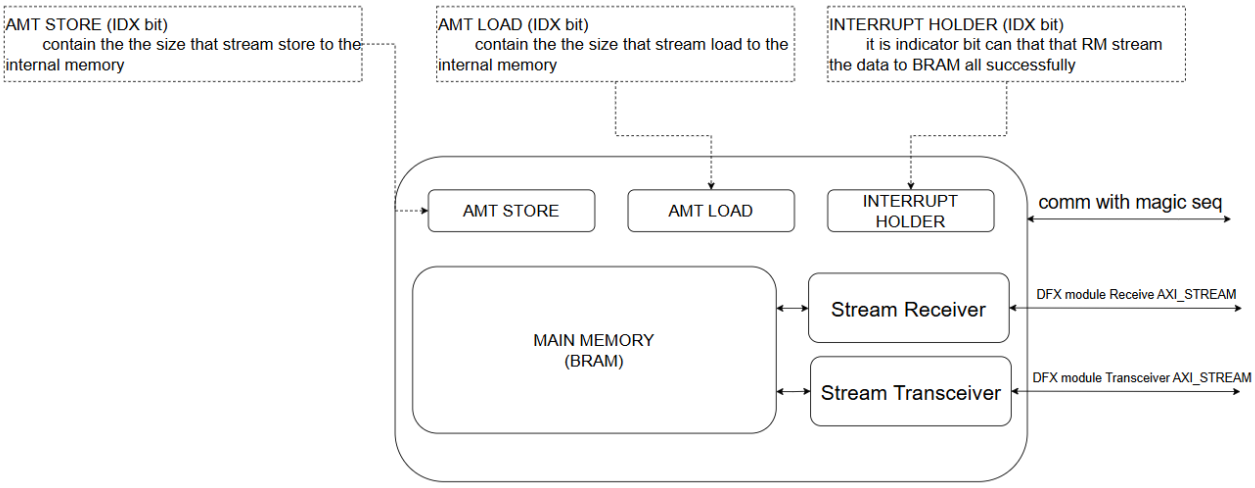


Figure 10 Magic Streamer Sub System Diagram

3. HLS4ML system design automation

To allow designers to use this feature, we integrate Magic Architecture from the previous section into the HLS4ML framework. The framework is responsible for chopping the Machine Learning model into multiple subgraphs. The produced subgraph is then analyzed to assign the proper connection to the matched magic sequencer. HLS4ML will then generate a special Verilog file that aggregates matched hardware for the current HLS4ML project. After HLS4ML has all the necessary hardware source files, it will invoke Vitis to synthesize each subgraph independently and Vivado to compose all hardware subsystems together.Finally, Vivado will generate the bit stream file, which is ready to ship for Hardware. (Figure 11)

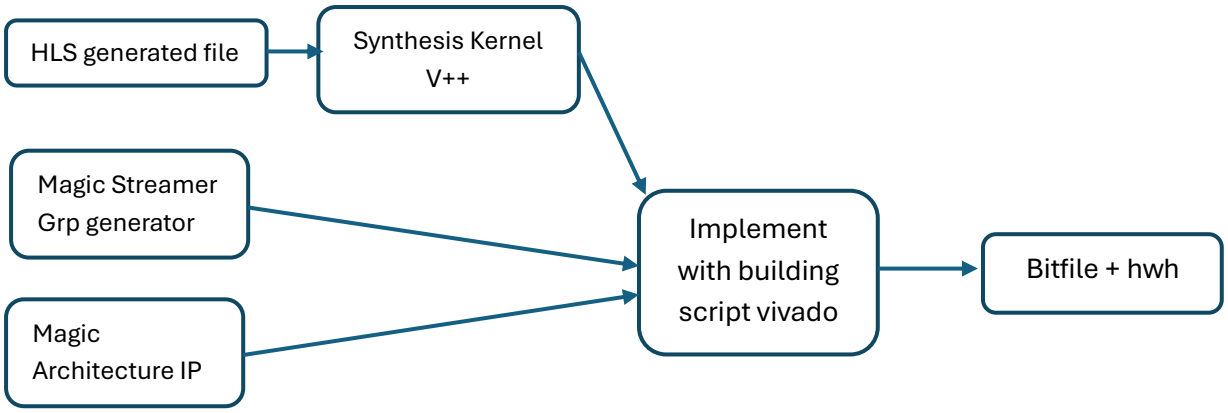


Figure 11 Vitis Unified Partial Backend

Testing

We started testing with a simple neural network layer, as shown in Figure 12. We divide the full model into three subgraphs. Each reconfigurable module (RM) consists of two fully connected layers, each comprising 100 input and output parameters.

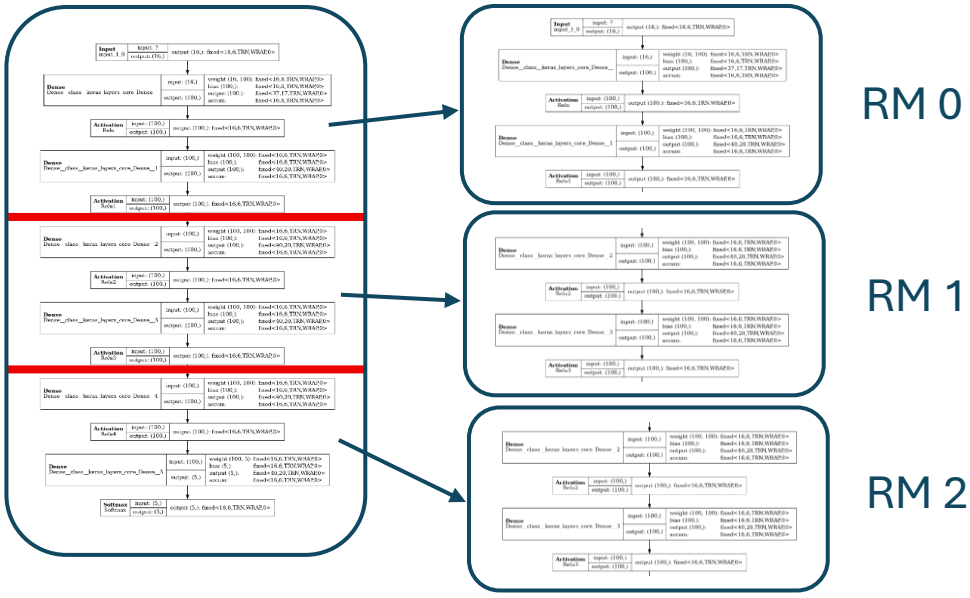


Figure 12 Reconfigurable Module Visualization

In this case, we did not use the HLS4ML automation workflow. Figure 11 Vitis Unified Partial Backend. We build dedicated hardware integration in Vivado, as shown in Figure 13.

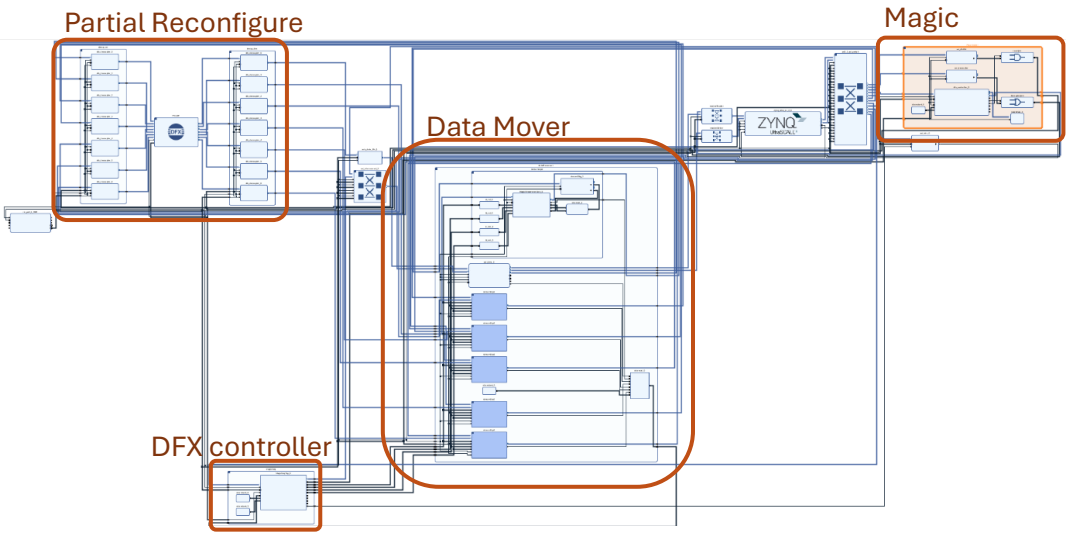


Figure 13 Custom Integration of Magic Architecture

1. Resource Usage

We then observe the resource usage for each resource type: lookup table, Flip-flop, and lookup table RAM, compared to the module without performing partial reconfiguration. In Figure 14The three left bar groups show the resource usage for each resource type for each reconfigurable module, and the far right bar shows the resource usage for the module without applying the partial reconfiguration technique.

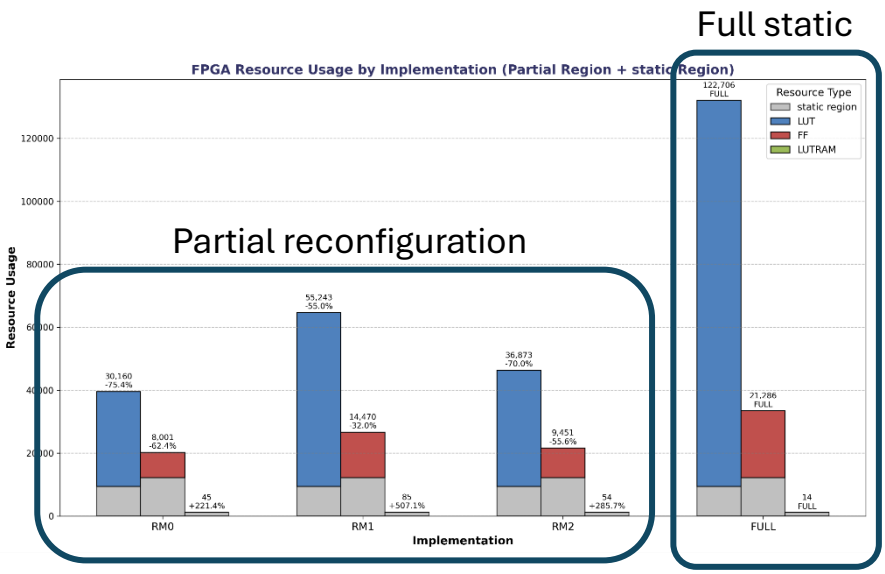


Figure 14 Resource Usage in Big Dense Keras

The results show that this technique has the potential to save LUT 55.0 – 75.4% and FF 32.0-62.4% compared to the static method. However, the system produces some overhead in LUTRAM resource usage for 31 – 40 cells.

Figure 15 illustrates the actual implementation for each module. The orange cells indicate the static cells, which are not dynamically reconfigured. These cells belong to DataMover (Magic streamers + DMA IP), DFX controller IP, and Magic Sequencer IP.

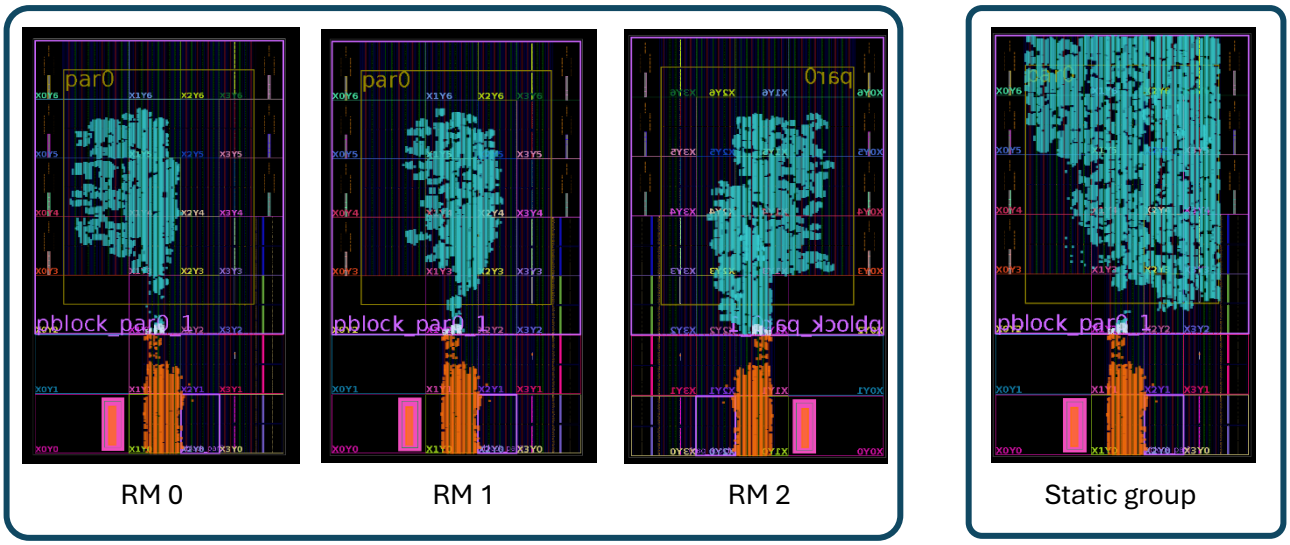


Figure 15 Fabric's cells implementation for each reconfigurable module and static use-case

2. Execution Performance

We measure the execution performance using the same ML model and compare the execution time in applying the partial reconfiguration method with the static method. However, the magic streamer is deprecated when the number of queries is over 675 due to the Bram limitation in ZCU102.

In Figure 16The blue line with a circle mask indicates the wall clock time from the partial reconfiguration (PR)method, and the blue line with a cross mask indicates the wall fabric reconfiguring time from partial reconfiguration using the profiler counter. If we consider the reconfiguration time as the system overhead time, the overhead is extremely significant when the number of input queries is small. In contrast, the overhead is less important when the number of input queries is larger. In conclusion, to optimize the execution performance of the PR technique, the designer should put the data and execution time usage for each reconfiguration module as much as possible.

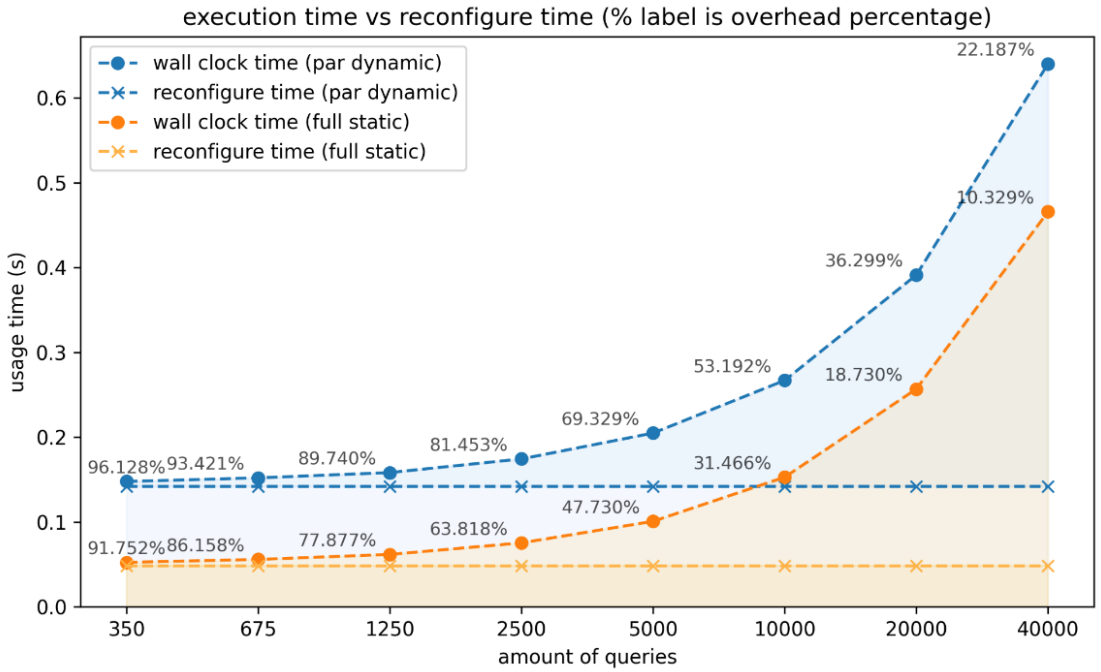


Figure 16 Execution time graph for different query sizes and implementation type

Similarly, the two orange lines represent the performance of the static method, the method without using PR. In comparison with the PR-based approach, the static approach demonstrates superior performance. However, despite this apparent advantage, static execution becomes infeasible when the machine learning model exceeds the available FPGA fabric resources.

3. Reconfigure Performance

We measure the performance of FPGA reconfigure time in ZCU102 by enabling the profiler counter integrated in the magic sequencer IP to count the number of waiting cycles. We use the simple Kerras model to compare the reconfiguration timing performance between the partial reconfiguration approach and the static approach.

Figure 17 shows the reconfigure time result from the profiler counter. The 3 bars on the left side demonstrate the performance in the PR technique, which is reconfigured by the DFX controller IP. The 2 bars on the right side demonstrate the performance in PR technique, which the FULL ICAP is reconfigured by the DFX controller as well, but the FULL PYNQ bar is reconfigured by the processor. In comparison, processor reconfiguring with DFX controller IP reconfiguring, the self-reconfigure using DFX IP performs 6.3 times faster. In addition, the self-reconfigure approach's performance is limited by ICAP throughput.

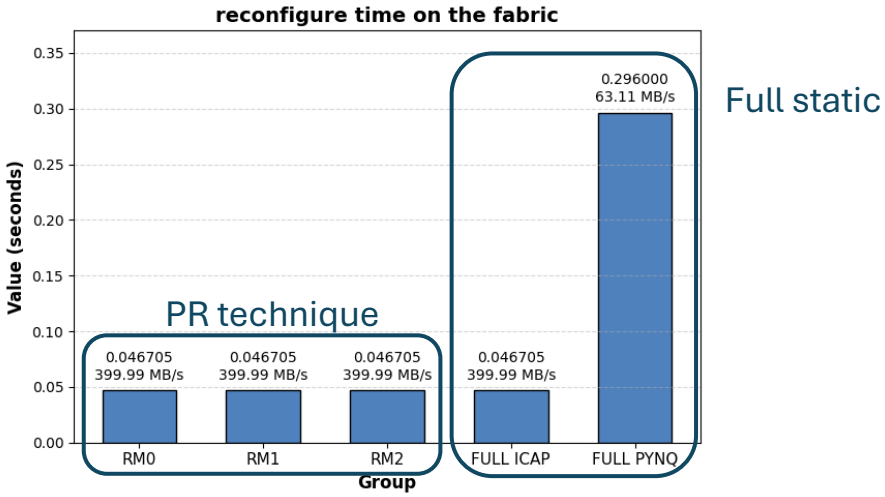


Figure 17 reconfigurable time for each reconfigurable module and implementation type

In comparison, FULL_ICAP in the static model and PR approach, the result suggests that the reconfigure time is not correlated with the number of cells in the module. In fact, the reconfiguration time depends on the number of allocated cells in the FPGA region.

Current Progress

Currently, we are attempting to deploy the U-Net model shown in Figure 18. Unfortunately, we encounter an overutilized problem. The HLS4ML generates too large an ML model, even though we have chopped it down into multiple subgraphs, as shown in Figure 18.

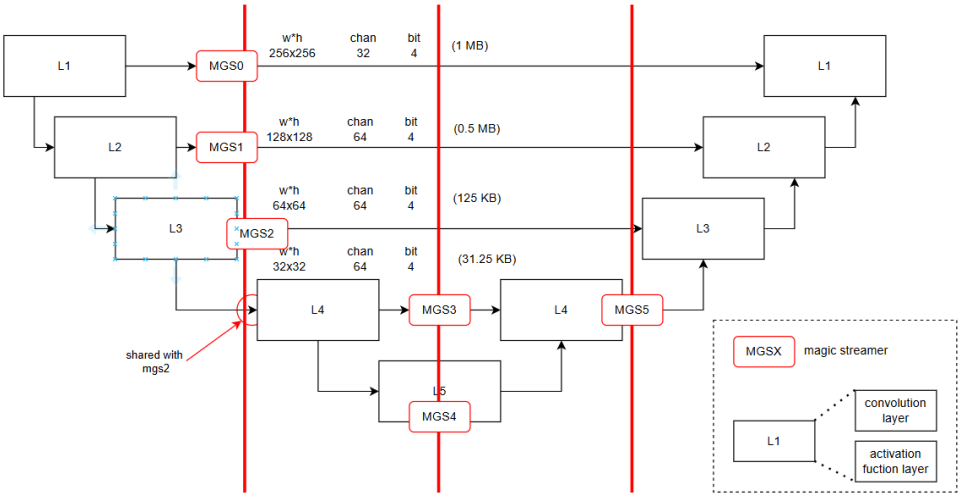


Figure 18 simplified U-Net layers

As shown in Figure 19Subgraphs 2 and 3 are not fit to the ZCU102 fabric; therefore, we then attempt to increase the reuse factor in HLS4ML. The reuse factor is supposed to reduce the hardware resource usage because some of the hardware resources will be reused to compute different data. Unfortunately, after we increase the reuse factor, the system consumes more resources than without reuse.

g4lm	constrs_1	write_bitstream Complete!	10.64	0.000	0.000	0.012
child_1_g4lm		write_bitstream Complete!	4.258	0.000	0.000	0.011
child_2_g4lm		write_bitstream Complete!	7.638	0.000	0.000	0.006
child_3_g4lm		place_design ERROR				
child_4_g4lm		place_design ERROR				

Figure 19 Sub Graph Implementation Error

After we increased the reuse factor from 1 to 8, all resources except Bram exceeded the base baseline (reuse factor = 1). We realize that bugs have occurred in the HLS4ML; therefore, we start breaking down the ML and analyzing layer by layer. Eventually, we have found that the depth-wise convolutional response has unexpected behavior.

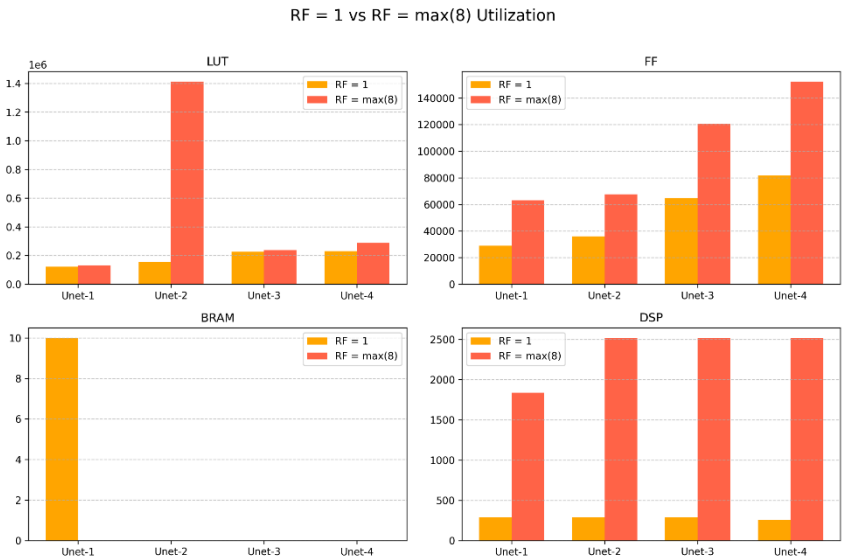


Figure 20 ZCU102 Resource Usage from Different Reuse Factor

The orange lines in Figure 21 show the resource usage of depth-wise convolution from original HLS4ML. It suggests that depth-wise convolution anomalously consumes Look Up Table (LUT) and FF (Flip Flop) resources.

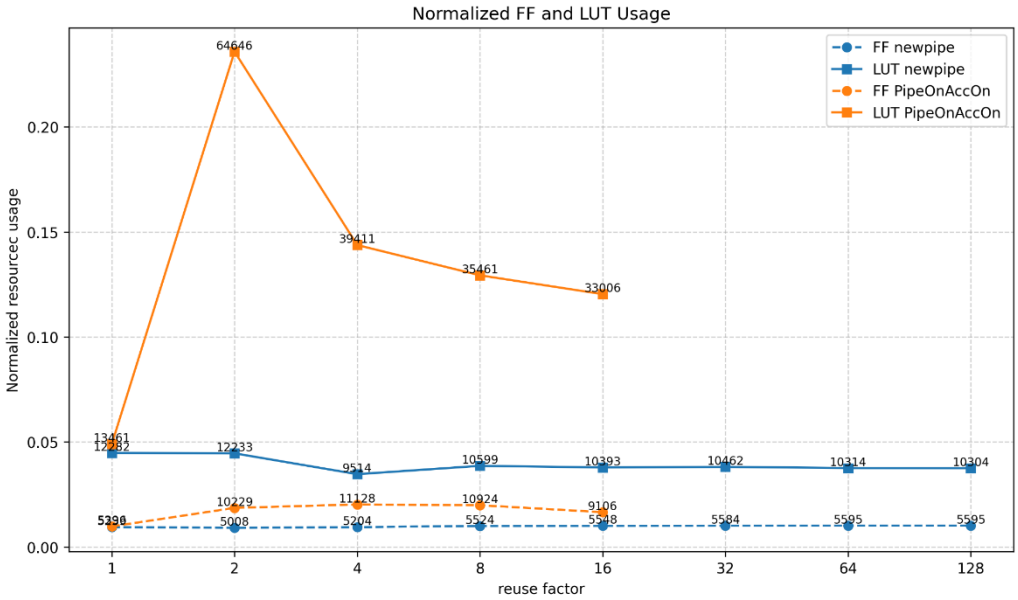


Figure 21 Flip Flop and Lut Usage in Different Reuse Factor and Configuration

Therefore, we try to modify the HLS4ML kernel by inserting a pipeline pragma at the convolution kernel iteration loop, as shown in Figure 22. According to the modification, the blue line Figure 21 show an acceptable, usual LUT and FF resource usage.

```
ReadInputHeightSerial:
    for (unsigned i_ih = 0; i_ih < CONFIG_T::in_height; i_ih++) {
    ReadInputWidthSerial:
        for (unsigned i_iw = 0; i_iw < CONFIG_T::in_width; i_iw++) {
            #pragma HLS LOOP_FLATTEN
            #pragma HLS PIPELINE II=CONFIG_T::reuse_factor
            if (CONFIG_T::filt_height > 1) {
                compute_depthwise_output_buffer_2d<data_T, res_T, CONFIG_T>(data.read(), line_buffer, res, weights,
                                                                            biases);
            } else {
```

The introduced line

Figure 22 Depthwise Main Loop in HLS4ML

Unfortunately, we still encounter a Bram resource problem. Since the magic streamer space requirement shown in Figure 18 requires ~ 50% of the BRAM in the FPGA fabric and must be allocated in a static region. In contrast, we desire the static region to consume the fabric < 35% to keep the partial reconfigurable region as large as possible. Therefore, we then introduce

a new fabric floor planning technique to concede some reconfigurable region resources to the static regions. The static region's cells in Figure 23 show that some static cells are placed into the dynamic region. The protruding part is Bram only.

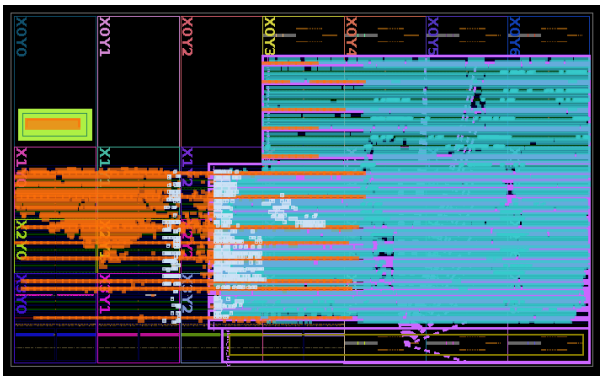


Figure 23 New Fabric Floor Planning Technique

Conclusion

This work introduces DFX4ML, a technique to exploit the partial reconfiguration mechanism for machine learning models. We start by extracting the full neural network graph into multiple sub-graphs. We also provide the magic architecture. The architecture controls the subgraph reconfiguring on the actual FPGA fabric, manages the inference dataflow, and cooperates with the processor. Moreover, we are still developing the VitisUnfiedPartial backend in HLS4ML to automate hardware integration progress. We tested the system with the big keras dense layer, the result shows significant resource reduction while it still maintains reasonable execution performance. In current progress, we are finalizing the VitisUnifiedPartial backend and attempting to deploy a practical U-Net model using the partial reconfiguration technique.

References

[1] “AMD Technical Information Portal.” https://docs.amd.com/r/en-US/ug909-vivado-partial-reconfiguration/Introduction-to-Dynamic-Function-eXchange

[2] FastML Team, “fastml_hls4ml,” *GitHub*. Zenodo, 2025. [Online]. Available: https://github.com/fastmachinelearning/hls4ml

[3] M. n. Abadi *et al.*, "TensorFlow: a system for large-scale machine learning," presented at the Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, 2016.

[4] F. c. Chollet and et al., "Keras," ed, 2015.

[5] A. Paszke *et al.*, *PyTorch: an imperative style, high-performance deep learning library*. Red Hook, NY: Curran Associates Inc., 2019.

[6] “AMD Technical Information Portal.” https://docs.amd.com/r/en-US/ug949-vivado-design-methodology/Vivado-Design-Suite-User-and-Reference-Guides

[7] “AMD Technical Information Portal.” https://docs.amd.com/v/u/en-US/ug1416-vitis-documentation

Preparation Diary Report

Friday, 4 April 2025

I learn how to install and build Petalinux and build an FPGA platform on my KV260, following this tutorial. Unfortunately, the Xilinx tutorial is a bit ambiguous, and Petalinux took a lot of storage space and compiling time.

11-13 April 2025

Unfortunately, while compiling the petalinux, the system provides an error about a PMU firmware error, and I spent more than two days solving this bug. I am not sure exactly where the bug comes from; however, the system runs the simple vector addition correctly.

23-25 April 2025

In three days, I am invited to participate in the Basic Particle Physics Training Program 2025 at Srinakharinwirot University. This program gives me a particle physics lecture. In fact, I don't understand it much because I do not have much foundation in particle physics. In this session, I attended the cloud chamber and particle identification workshop.

Monday, 12 May 2025

Today, I attempt to build a custom DMA design for KV260 using Vivado 2023.2. We then ask Vivado to generate a (.XSA) file to put it in the Petalinux for the operating system generation. Actually, it is not an efficient way because I have to rebuild the system for many hours, even though we have changed only a little of the hardware design.

13-16 May 2025

Unfortunately, while compiling Linux, the compiler crashed. The previous problem hit me again. I tried many configurations to fix this problem. Fortunately, the compiler works correctly, and I have found that if the connection specification is not set when creating the Vivado project, the system will crash. Finally, we can test the provided DMA perfectly.

Saturday, 17 May 2025

Today, I start to learn how PS-PL communication works. In this process, I tried to make the system run without an operating system and operate using barebones software. We do this because I aim to reduce the testing time by skipping the petalinux compiling process.

Sunday, 18 May 2025

Unfortunately, the barebone does not work because I can't override the KV260 firmware to not search for the operating system. We return to using petalinux instead. We then start to recreate the platform and deploy it on an FPGA.

Monday, 19 May 2025

I attempt to test the system by using DEVMEM. DEVMEM is the command-line program embedded in PetaLinux, so I can send the data to the FPGA fabric. The test was successful. I can send data from the KV260 processor to the DMA IP in the FPGA. Moreover, I watched this video to learn high-level synthesis programming ([Tutorial] Productive Parallel Programming for FPGA with High Level Synthesis)

Friday, 23 May 2025

I start learning how to enable dynamic reconfiguration using custom partition declaration from the Xilinx document, and attempt to build my own dynamic function exchange on Vivado.

Saturday, 24 May 2025

I start to export the platform from Vivado; however, the system shows that "can't find dynamic bd". (ERROR: [Project 1-1039] Failed to get Composite File object for dynamic region BD when exporting DFX hardware platform.) I fix it by excluding the dynamic region and building a platform (.xsa) from the separated region.

Sunday, 25-26 May 2025

I started to compile the Petalinux again, and it completed successfully; however, the system gets stuck when booting. Eventually, I have found that the error comes from the AXI GPIO driver IP. After I manually delete the GPIO driver. The system works fine.

Monday, 27 May 2025

I start to custom-build Bram and the Bram controller. I alter the Bram output port data and let the logic that alters the output data be placed in the partial reconfigurable region. Finally, I use the DEVMEM to communicate with the Bram, and the system works correctly when testing.

Tuesday, 28 May 2025

Using DEVMEM to set physical memory is quite dangerous. DEVMEM accepts the command from the user. If the user assigns the wrong address, the program can write in the wrong place, and the OS can permanently collapse. Therefore, today I learn to write a user space program using the C++ language.

29 - 31 May 2025

I started to focus on High Level Synthesis programming because the HLS4ML project is truly based on High Level Synthesis. I start learning how to control the system using AXI Lite and investigate how to collaborate on driver generation.

Sunday, 1 June 2025

Today, we successfully deployed partial reconfiguration using my own generated IP.

5 – 7 June 2025

These 3 days, I start breaking down HLS4ML to see the internal structure of the system. I mainly focus on how the neural network model is generated.

Diary Report

Tuesday, 10 June 2025

Today marks the first contract day, and the summer students are required to attend the first meeting session. In this session, we are introduced to the activities offered at CERN and receive a debit card, through which our living expenses will be transferred. Specifically, they introduce us to the safety protocols that we must comply with at CERN. After completing this session, we are scheduled to meet with our supervisor for the first time. We have discussed the project, and we are booking the working desk for the entire summer.

Wednesday, 11 June 2025

I woke up at ~6.00 a.m. The weather was a bit cold. My first task was to build the PYNQ framework + for hardware execution. The framework is the program that cooperates with Petalinux to allow hardware designers to customize hardware design without rebuilding any os. To build PYNQ for ZCU102, I started by installing a virtual box on my laptop. Just for the starting step, there are a lot of problems, such as secure boot blocking, vagrant not supporting, and building blocks. I fell asleep at my accommodation at 10.00 p.m. while my laptop was compiling pynq + petalinux all night.

Thursday, 12 June 2025

I woke up at ~6.00 a.m. The weather was a bit cold. After compiling all night, we got an error about the YOCTO layer. It occurs because my compiling settings are not correct. After some modifications, I started to compile it again, and it compiled successfully, as shown in Figure 24Those are the files that we needed.

After my supervisor found out about the compilation, he asked me to test my system at his lab. There is a bit of a problem with internet access. That is, the system cannot be accessed via SSH. We figure it out by locking the MAC address of the board. Then, I started the next task, which was to design the hardware to support hardware partial reconfiguration.

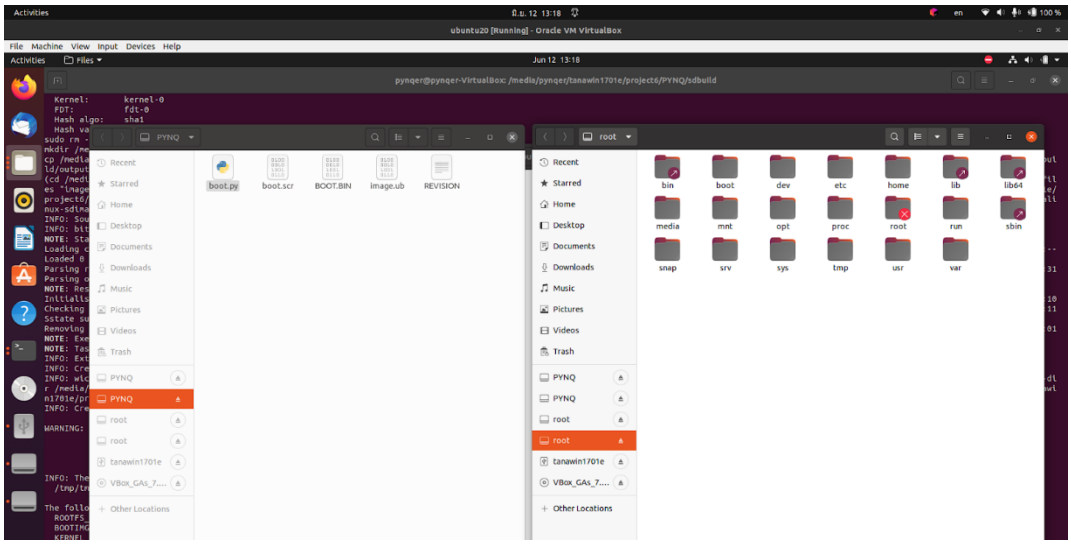


Figure 24 Ready to Ship File for ZCU102

Friday, 13 June 2025

After waking up at 08.30 a.m., I received the lab meeting from my SPA lab in Thailand with Assoc. Krerk Piromsopa. The session is 30 minutes long and mainly talks about the thesis update project.

After having breakfast, I continued the hardware chip design to support partial reconfiguration. I started with doing simple vector addition hardware, as shown in the picture. Figure 25.

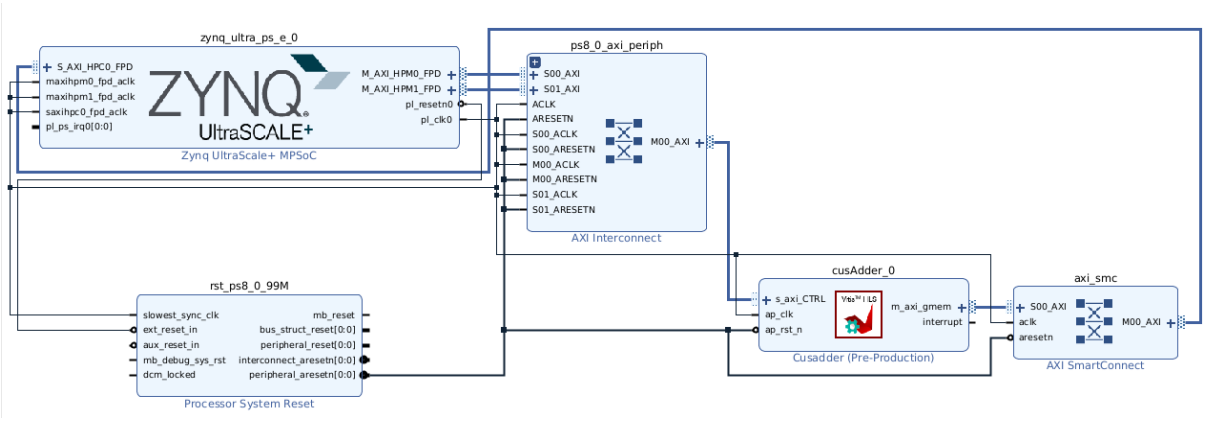


Figure 25 Vivado Integration Diagram for Simple Vector Addition

Unfortunately, the dumped result was unexpected. That is, two positive vectors are performed in the hardware chip, and the results are sent back to the main processor; however, the result is falsified from our expectation.

Until 3.30 p.m., I found that the results are altered if the two vectors share the same data loader port. Therefore, I have managed by providing a dedicated port. Next, I integrated the fixed hardware to be a partially reconfigurable chip.

Saturday, 14 June 2025

From 0:00 to 3:00 a.m., I do the laundry in the basement of Building 38. At 7:00 a.m., I wake up, fold the pile of clothes, and have breakfast; it consists of salmon cheese cake and a brown chocolate bread. After that, I worked with the partial reconfiguration from the last day, and the expected result was populated around 1.00 p.m. That triggered me to have a happy lunch.

Around 2.00 p.m., I continued upgrading the design to equip it with an interrupt controller because, for now, we can send the command from the processor to the target chip, but it cannot send the command(interrupt) back to the processor.

Until 10.00 p.m., the restaurant personnel told me that the restaurant was closing. At that time, I can get the interrupt signal back from the processor. The design is in Figure 26.

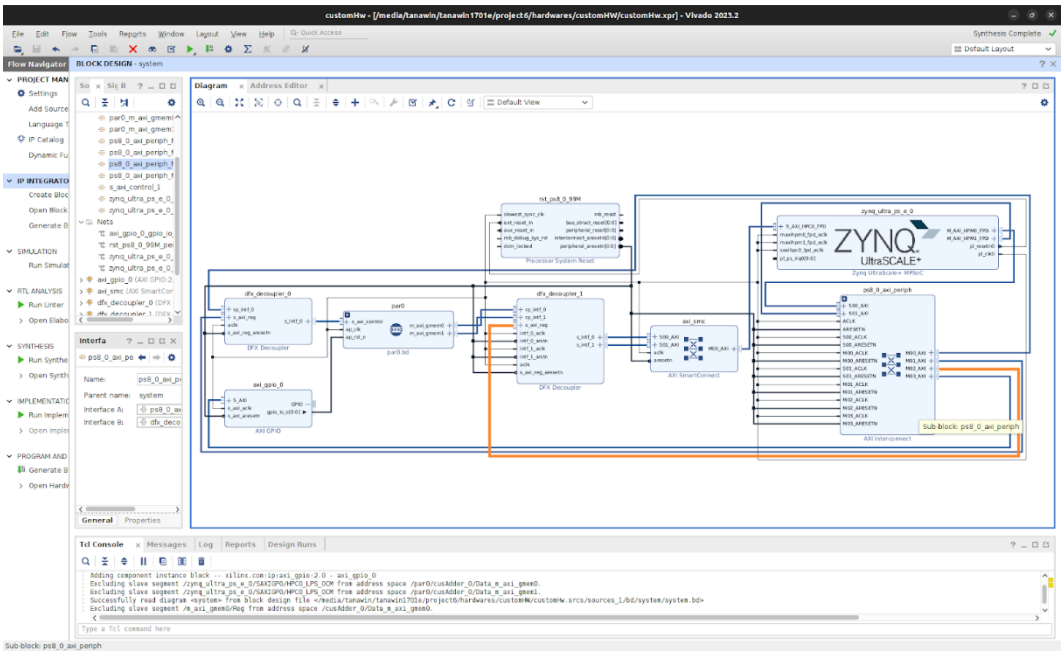


Figure 26 Interrupt System Block Design

Sunday, 15 June 2025

I start to continue my work at 08.00 a.m. to upgrade and check the interrupt system at the restaurant. Unfortunately, at 11, I forgot to re-couple the hardware; as a result, the FPGA board got stuck. The decoupler is shown in the red rectangle at Figure 27.

The only way to resolve this situation is to physically reset the board, but it was locked until my supervisor is back on Tuesday. In the meantime, I had to complete the document work instead. Moreover, a strange student requested advice from me to solve the data transfer between the Server.

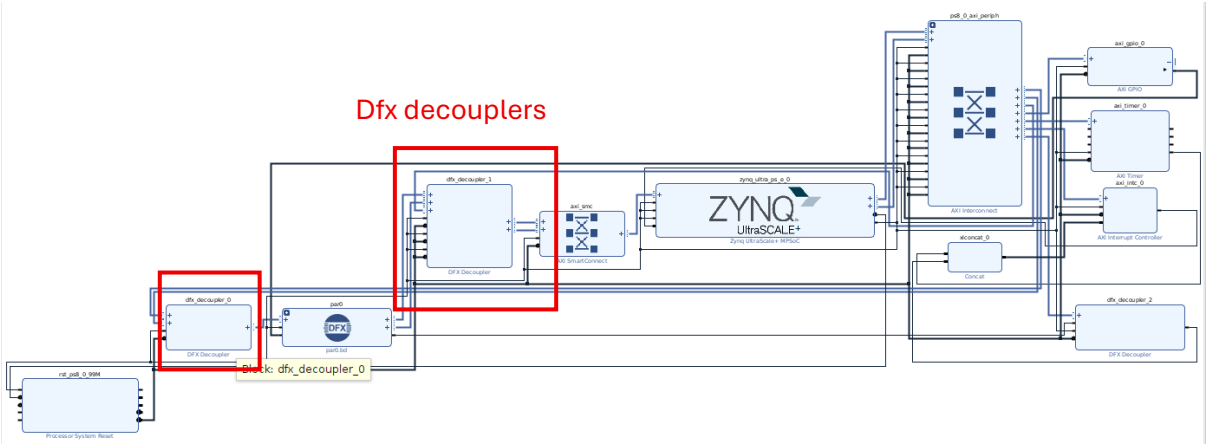


Figure 27 the Vivado Block Diagram and the Decoupler IP in the Red Box

Monday, 16 June 2025

I woke up at 05.00 a.m., and the weather was slightly cold. After I have breakfast at restaurant 1, I continue my work in my room again. The primary purpose of today's work is to prevent the system from getting stuck again. I drafted the new method to automatically re-decouple the system when the hardware was invoked. In the afternoon, we have site visits. That is the ATLAS accelerator and synchrocyclotron. We experience the control room of the ATLAS and its purpose. Moreover, at Synchrocyclotron, we receive the history of CERN and what CERN archives. This day is very exciting!

Tuesday, 17 June 2025

I woke up at 06.00 a.m., and the weather was slightly cold. Today I had a meeting appointment with our lab member to update my work progress. The feedback was very delightful because I can enable the partial reconfiguration in the first week! In the afternoon, I completed the interrupt system and went to Carrefour to fill the supply. At the carrefour, I almost can't pay for the goods because there is an error with my CERN debit card. In conclusion, I have to pay with my SCB debit card.

At night, I come back to work again at my accommodation; however, we now focus on the HLS4ML, the hardware framework instead.

Wednesday, 18 June 2025

I forgot when I woke up; however, the weather was good as it was. Today, I'm very excited because there is a data center site visit. In the meantime, I started the machine learning part (HLS4ML) tutorial. The purpose of this session was to encourage myself to get familiar with the framework. In addition, this framework will be what I make it have reconfigurable capability. In the Afternoon, I went to a site visit meeting point, the restaurant, and then we were guided to the data center. That was pretty. Eventually, I saw a huge data center for the first time in my life. Moreover, that place exhibits the old computing and networking devices to show how CERN has managed the data over 70 years!

At the end of the day, my supervisor texted me that the office reservation is confirmed! The desk is in building 653. Even though it is a bit far from my accommodation, it is quiet and has an air-conditioner. In CERN, I think that it was rare to have an air-conditioner, and it's essential for the next two months.

Figure 28 CERN Data Center (left), Old used Computer at CERN (right)

Thursday, 19 June 2025

I woke up at ~6.00 a.m., the weather was good as it was. Today, I continued the HLS4ML all day until 8.00 p.m. In addition, it was the first day that I made lunch for myself, and it was soaked salmon served with bread that I brought from Carrefour. I took the tutorial until 8.00 p.m. After having dinner at a restaurant, I got back to my bedroom and continued the tutorial until 10.00 p.m.

Friday, 20 June 2025

I woke up at 05.00 a.m., and the weather was a bit hotter. I go to the office at ~7.45 a.m. and continue the last part of the tutorial. The last step was to deploy the generated machine learning to the hardware. However, there is an obstacle, that is, the system was built for the 2020 version, which is not currently supported. I must export the hardware using an alternative approach and modify the code to support my current version. Fortunately, my supervisor experienced this problem before and suggested that I use the unofficial version. That was very helpful. Moreover, he provides me with a new monitor. That changes my whole work life quality! Today's dinner was at 7.00 p.m.; it was rice and some sort of fried ball that tasted like curry. I then get back to the office again and finish my last tutorial section at 10.00 p..m.

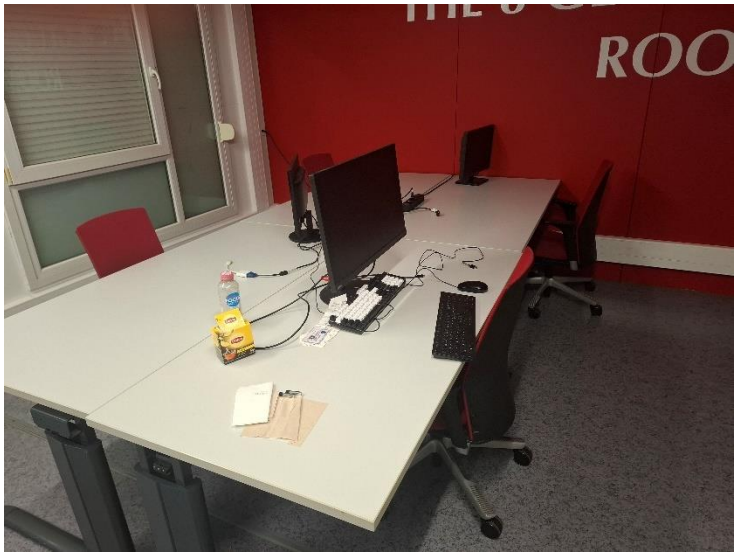


Figure 29 My monitor that I used at CERN

Saturday, 21 June 2025

The weather was fine. I go to the office at 8.00 a.m. and start to finalize the HLS4ml tutorial from yesterday. Fortunately, this task works fine. Therefore, in the afternoon, I start to do the next task, which is to enable the FPGA to write the chip by itself. This idea is very exciting. I started to explore the Xilinx document and tutorial, and started to customize my own design for the ZCU102 board by myself. Finally, I get back to my room at 10.00 p.m.

Sunday, 22 June 2025

Today, we had a hostel room transfer. I have to transfer from a single bedroom type to a double twin bedroom type. I feel a little bit sad because I love to stay private. I start to pack my

luggage and check into a new room. At 11.00 a.m., I started to continue the self-FPGA reconfiguration again. By the way, at 1.30 a.m., I get back to the hostel to make my soaked salmon for lunch and pack it for dinner. Finally, I got back to the hostel at 10.00 p.m.

Monday,23 June 2025

I woke up at 5.30 a.m. and had lunch at the restaurant 1. The pineapple pie was yummy! I started to work on the FPGA until 11.00 p.m. Unfortunately, I have found that it is impossible to do the FPGA's self-reconfiguration because the secure boot of the chip prevents the developer from accessing the essential hardware that allows us to configure the important part of the device. The only way to manage this situation is to disable the chip's security system. By the way, at the restaurant, I had experienced a good vegetarian soup with fried.

Tuesday, 24 June 2025

I woke up at ~6.15 a.m. and had lunch at restaurant 1. Today, the weather was good. I go to my office at ~8.00 a.m. There are two main tasks for today. The first is to create the presentation that I did last week. This task is pretty easy; however, the second task was extremely hard. That is to figure out how to enable the FPGA (Field programmable gate array) to accept my reconfigure command. Unfortunately, at the end of the day, I still can't figure it out, but I realize that the boot file must be modified to fix this problem. Therefore, at 5.15 p.m., after updating the progress to my supervisor, I went to his office to modify the boot file. Luckily, I can solve this problem, but the result is still wrong. Finally, I got back to my accommodation at 10.00 p.m.

Wednesday, 25 June 2025

I woke up at ~5.15 a.m. I build my own breakfast and lunch, which are rice porridge and instant noodles with chicken on top. I go to my desk at ~7.30 a.m. The weather is fine; however, because my office has a shared air conditioner, it makes me cold! Now, I am building a new hardware design to check why the system cannot perform as it should. Eventually, I figured out that the bitstream file used to configure the system was falsified by the numpy library. After this success, I created bigger hardware designs to ensure that the self-reprogram technology works well. Finally, I get back to my accommodation at 10.00 p.m.

Thursday, 26 June 2025

I woke up at ~5.15 a.m. I have breakfast at restaurant 1. I go to my office at ~7.00 a.m. The weather was a bit cold, and my office mate still turned on the air conditioner again.

The task for today is to apply the HLS vector adder and subtractor to the partial reconfiguration design. In addition, the purpose of this task is to mimic the HLS4ML workflow design. At ~5.00 p.m., I finally achieve the result from the design; moreover, I continue with the bonus task. The task is to make the hardware chip (FPGA) reconfigure itself. This is a very interesting idea! It sounds like “A surgeon performs surgery on himself”. Finally, as on other days, I get back to my accommodation at 10.00 p.m.

Friday, 27 June 2025

I woke up ~05.15. The weather is good, and I have my favorite breakfast. That is pineapple pie. I go to my office at 07.00 a.m. I start to design the internal structure of the self-reconfigurable chip. According to the task “make it auto reconfigure when the first task is finished”, I generalized my design to accept an arbitrary number of tasks and make it communicable to automatically resume the task. After drafting the design, I started writing the Verilog code by myself from scratch, which is pretty heavy. I spend time with it until 10.00 p.m., but I am still not finished. I get back to my room at 10 p.m.

Saturday, 28 June 2025

I woke up at ~5.30 a.m. The weather is a bit cold. After that, I immediately go to the kitchen to make an instant breakfast and pack the instant noodles and the soaked chicken. Then, I go to my office at ~8.00 a.m. to start finalizing the code that I had drafted yesterday. But not to rush, I only built half of the project, and tested only half first. The reason that I don’t build the whole system at once is that if we deploy the hardware and there is an error, we cannot know where the error is. At ~ 06.00 p.m., I start to write the driver for the board to control and interact with my design. I get back to my accommodation at ~10.00 p.m.

Sunday, 29 June 2025

I woke up at ~6.00 a.m. The weather was a bit hotter, I had breakfast at restaurant 1, and went to my office like the previous day. Today's task is to test the driver and hardware that I wrote about yesterday. Fortunately, the system is working well, so I continue to complete the second half. This part is harder because it involves many types of hardware, and the document is pretty long. At ~6.30 p.m. I have a dinner at restaurant one and will get back to continue my work until 9.00 p.m. After that, I stroll around the Meyrin site. The weather is pretty fresh, and the environment is pretty peaceful. On the west side, there is much forest and fewer buildings. I love this moment very much. Finally, I get back to my accommodation at 10.30 p.m.

Monday, 30 July 2025

I woke up at 5.15 p.m. The weather is a bit hot, I have breakfast at a restaurant like the other day, and I arrive at my office at ~ 7.30 a.m. I try to write the driver for my whole system. Unfortunately, the hardware got stuck somewhere. I will try to solve this problem all day. Finally, I found that the error came from my mistake about the module restart setting. Finally, after having the solution, I started to build a presentation for tomorrow's meeting. At 10.00 p.m., I get back to my accommodation and get a good sleep.

Tuesday, 1 July 2025

I woke up at 6.00 p.m., and I went to my office to finalize my presentation slide. Today, the first lecture programme starts at 9.15 a.m. The lecture introduces us to CERN, the lecture program, and the OpenLab session. The OpenLab schedule interests me very much because there is FPGA and GPU programming. I have tried to learn these topics many times, but I don't get it. I hope this session will be a good starting point for me. However, I ditched the last session because I have a meeting with my lab. I think my presentation works well! I get back to my office in the afternoon to continue my work to upgrade my hardware. Until 9.00 p.m., I get back to my accommodation.

Wednesday, 2 July 2025

I woke up at 6.00 a.m. The weather was a bit hot. I went to my office at 7.30 a.m. and tried to test the Machine learning graph divider. This is the essential part of my project. That is, the machine learning that will be deployed to the hardware is divided into multiple parts. We do this because we need to save space on the chip by making it reprogrammable by itself. This is a very interesting idea. However, there are some bugs in the HLS4ML frameworks. I have to break the framework down to make the system work. By the way, restaurant 1 announces that they made pad thai for the main course. At first, I thought it would be great to have Thai food for lunch. Unfortunately, that is not Pad Thai; it tastes like something salty and sour spaghetti. They disappoint me very much. Today, I feel tired, so I got back from the office early (9.00 p.m.)

Thursday, 3 July 2025

I woke up at 6.30 a.m. The weather was good. I went to my office at 7.15 a.m. The task for today is to build more tests on the HLS4ML framework with the hardware partial reconfiguration technique. I modified the driver all morning. At lunch time, I have a soaked traut with grain. This

launch was a lot better than yesterday. In the afternoon, unfortunately, when I tried to run my driver, the system crashed, and I could not access the board at all. I am very disappointed in myself. Moreover, the board can't be reset until next Tuesday. Now, I am completely behind schedule. However, I will not give up; I tried to do other tasks that will be integrated with my system, such as an automatic model builder, to fill this gap in time. By the way, today, I went to fill up my supply. After filling up the supply, I get back to my office and continue working until 10.30 p.m.

Friday, 4 July 2025

I woke up at 6.30 a.m. The weather was OK. I went to my office at ~7.30 a.m. My driver crashed. Fortunately, my supervisor told me that his collaborator will reset the board at his office. At 11.30 a.m., I eventually was able to get access to the board again. After I had launched, I rewrote the drive more concisely. That is, I make the driver ask me every time it intends to access the hardware. When it is asked, I will check the accessing address and the accessing data. Finally, at ~19.00, the system can execute the driver without getting stuck. Unfortunately, the result is not correct. I tried to figure out the problems until 11.00 p.m., then I got back to my accommodation.

Saturday, 5 July 2025

I woke up at ~6.00 a.m. The weather was a bit cold. I had a pineapple pie for breakfast at restaurant 1. I went to my office at ~8.00 a.m. The main goal for today is to fix the hardware that produces the invalid input. I therefore revised the Xilinx DMA manual. It is a manual of the provided Direct Access Memory (DMA). The DMA is responsible for retrieving data from memory to my custom hardware chip and from the chip back to memory. Fortunately, I found that I forgot about the interrupt signal. This signal is crucial for my system, but I forgot to reset this signal. Eventually, at 1.00 p.m., I get back to eat a salad at the restaurant 1. After that, I take a tram with my summer student friend to go to the Geneva airport to pick up a high school teacher that joins cern high school teacher programme. After I get back to CERN, I go to my office at 4.00 p.m. to continue my work until 10 p.m.

Sunday, 6 July 2025

~6.00 a.m. The weather was a bit cold. I had breakfast at restaurant 1 like yesterday. Today, I did the laundry at ~6.30 a.m. In the meantime, I try to create my automation script for my partial reconfiguration project at restaurant 1. When my timer alarms that the washing machine is

finished. I go to the laundry room; however, it is still not finished. I don't know what is going on with this machine, so I was waiting at the building 39 lobby for 30 minutes. After finishing with the clothes dryer, I go to my office at ~10.30 a.m. to continue my work. At ~ 7.30 p.m., I have dinner at restaurant 1 and continue working there until 9.15 p.m. Then, I went running around the Meyrin site. The route is pretty impressive, the road is very steep, and the view is very beautiful. The weather is much cooler than inside the building.

Monday, 7 July 2025 ~6.30 a.m.

The weather was pretty cold. I had breakfast at restaurant 1 like yesterday. I go to the office around 8.00 p.m. The task for today is to investigate the problem and the possibility of doing the partial configuration on the hardware with the U-Net model. The U-Net model differs from other machine learning models. That is, the U-Net model does not send the data fully sequentially. It sometimes bypasses the data over the system; therefore, it makes multiple outputs for the partial reconfiguration partition. In the afternoon, my supervisor asked me to make the progress slide.

Tuesday 8 July 2025

I woke up at 6.15 a.m. The weather was pretty cold. I have breakfast at restaurant 1. I go to my office ~7.30 a.m. I started to fix my presentation slide, which was commented on by my supervisor yesterday. The main fixing point was to make the partition between partial reconfiguration and static plot. Then, I went to a meeting at building 42 at 10.00 a.m. The presentation got good feedback and received a new task. The task was to enable the partial reconfiguration in the U-Net model. I resumed my work at my office until 7.30 p.m. Then I left for restaurant 1 and resumed my work there until 10.00 p.m.

Wednesday 9 July 2025

I woke up at 6.30 a.m. The weather was pretty cold. I have breakfast at restaurant 1. I go to the office at ~7.30 a.m. I started the task that I received yesterday. The task for today is to deploy the hardware with the ONNX model. Unfortunately, the model is larger than the FPGA fabric. My supervisor advises me to use the Hardware Buffer Optimization technique to reduce the hardware resource requirement. Even though the technique is heavily applied to the ML model, the resource is still overutilized. At ~ 4.00 p.m., my supervisor told me that the ML model may not be possible to fit the hardware, and he suggested that I skip this task. I then skip this task and do the next task immediately. At 9.30 p.m., I went out jogging around the Meyrin site with my summer student friend.

Thursday 10 July 2025

I woke up at 6.00 a.m. The weather was less cold than yesterday. I went to my office at 7.30 a.m. The task for today is to modify the HLS4ML framework to support multiple IO in any ML. The issue is that the framework was written in the Python programming language. It is hard to do reverse engineering. In addition, the framework dynamically builds the type of hardware representation at runtime. I can't just know by reading the static code. Fortunately, I am used to the Python debugger, which enables me to see what is going on while it's running, at 7.00 p.m. I had dinner at restaurant 1 and then I resumed my task until 10.00 p.m.

Friday, 11 July 2025

I woke up at 6.30. The weather was like yesterday. After having breakfast at restaurant 1, I continue to modify the HLS4ML framework to support multiple IO at the desk near restaurant 1. Unfortunately, I spilled a lot of drinking water on the desk; therefore, I had to try to communicate with the housekeeper in French to get a tablecloth. At 8.50 a.m., I go to the CERN tram stop and wait for my summer student friends and Professor Norraphat. At 9.14 a.m., I started the journey to the Thai embassy. We had a meeting with the Thai ambassador, talking about our work at CERN. After that, we got back to CERN, I continued my work ~ 1 p.m. until 6.00 p.m., and I had a meeting with the HLS4ML team. This is my first time presenting with the global team. I feel a bit nervous, and got a deep question from someone at Caltech.

Saturday, 12 July 2025

I woke up at ~6.30 a.m. The weather was a bit cold. I went to my office at ~8.30. I continued to modify the HLS4ML framework to support multiple IO like yesterday; however, today's modified code is very promising. The code can run on the Vitis HLS program to synthesize the hardware model. There are two main portions in the framework that have been modified or created. The first part is the VitisAcceleratorIpPartial backend. It is responsible for converting

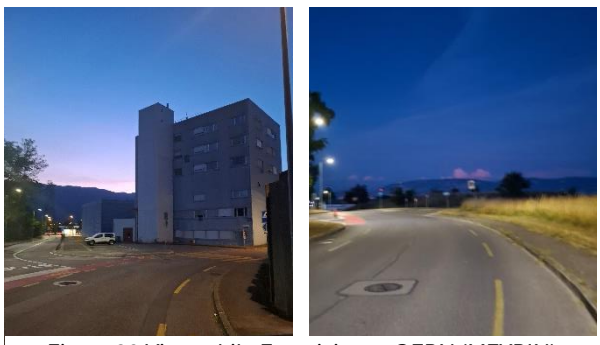
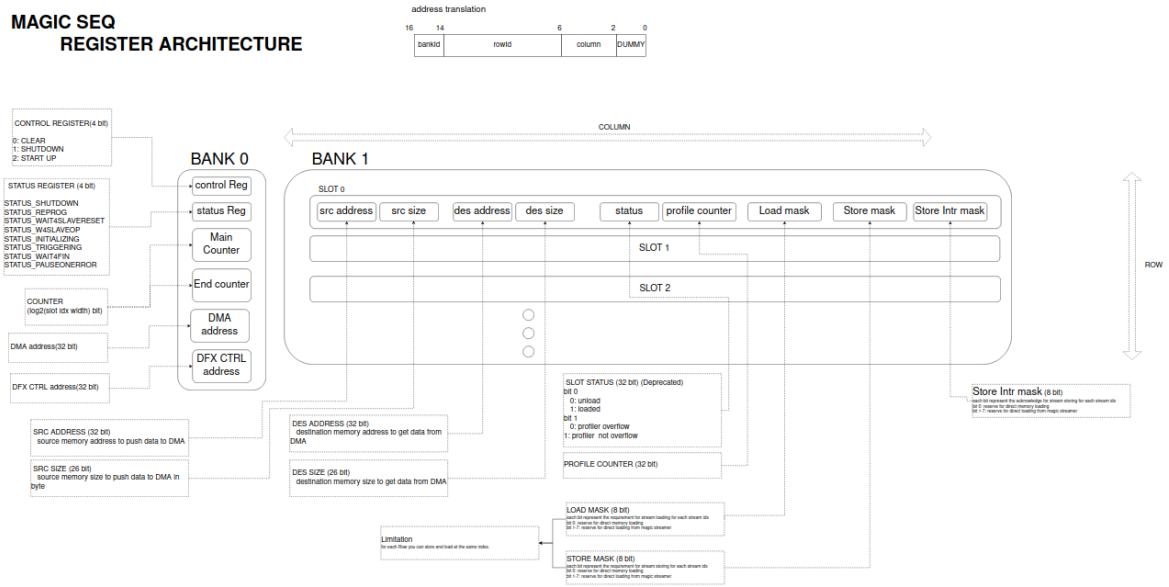


Figure 30 View while Exercising at CERN (MEYRIN)

intermediate models into high-level synthesis language. I had to create this one on my own to support multiple IO for each AI model. The second part is to modify the Multigraph class to support skip connection detection for the U-Net machine learning architecture, at 9.00 p.m. I do the laundry in the basement of building 39. While the washing machine was running, I went running around CERN (Meyrin). The weather was really cold there, while running, the sky was gently dark. When I headed back from the farthest west entrance point, the sky was dark and the route was forlorn. I arrived at my accommodation at 10.30 p.m.

Sunday, 13 July 2025

I woke up at ~7.30 a.m., and I felt tired from yesterday's running. I had breakfast at a restaurant the other day, and I went to my desk. The task for today was to upgrade the hardware to support the newly generated high-level synthesis code. To support multiple IO, we have to create a buffer to temporarily store the data. I called them “magic streamers”. Actually, my hardware block used to control the reconfigurable module is called a “magic sequencer”. Moreover, I not only modify the magic streamers but also upgrade the magic sequencer to control and track the status of the streamers. I left my desk at ~ 6.30 p.m. for dinner. After that, I continued my work at the restaurant until ~9.30 p.m. and then I headed back to my accommodation.



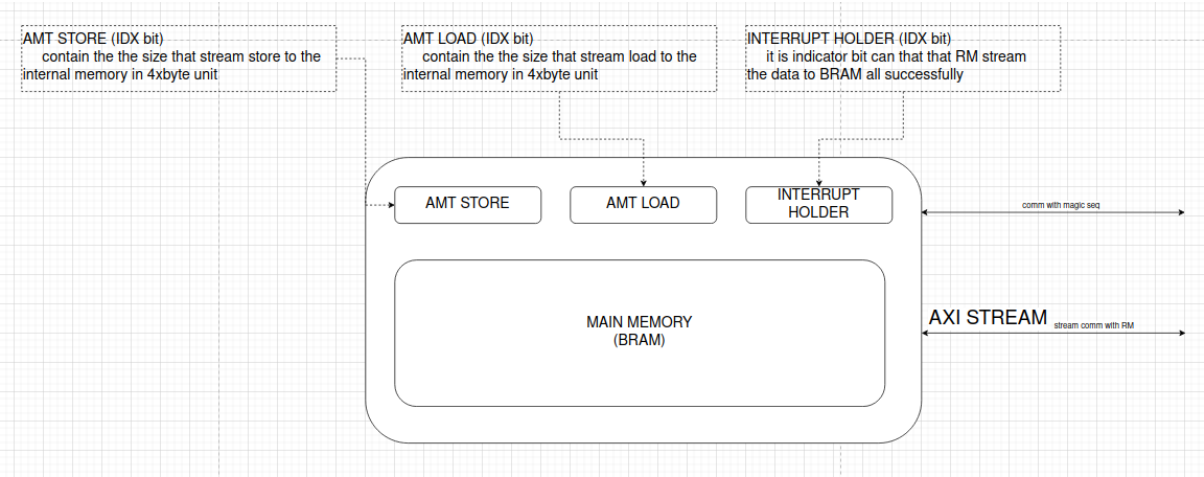


Figure 31 Magic Streamer Diagram

Monday, 14 July 2025

I woke up at 4.45 a.m. The weather was cold. I made breakfast in the kitchen and packed soaked salmon for lunch. I go to my office at 7.30 a.m. The task for today was to revise the code written yesterday, compose it with other subsystems in the Vivado program, and test it. Unfortunately, when I composed it in the Vivado program, I tried to synthesize it. The system was stuck with some errors. I don't know what happened to it. Therefore, I have to rebuild it from scratch. It means that I WASTE my morning on this error. I am pretty sad about it. I continued to work in the afternoon until 9.30 p.m. And I get back to my accommodation.

Tuesday, 15 July 2025

I woke up at ~6.00 a.m. The weather was a bit cold. I had breakfast at ~6.30 a.m. and then worked there too. The task for this morning is making the presentation slide for today's lab meeting. This is my presentation slide (https://docs.google.com/presentation/d/1ncXNhVfGJSJwoBSOCet1cMqj4AcvyYFR94TzUkK_TIo/edit?usp=sharing). It mainly talked about the modified architecture of the magic streamer and magic sequencer. At the meeting, my supervisor suggested how important it is to hold the data of the machine learning parameter on the FPGA fabric instead of sending the data back to the main memory. The main benefit of holding is to minimize the energy consumption and the latency. After having launched, I headed back to my desk to upgrade my design for making the memory loading/storing discardable. I worked until 9.15 p.m. I started running around CERN (Meylin site). Finally, I get to sleep at ~10.00 p.m.

Wednesday, 16 July 2025

I woke up at ~6.30 a.m. The weather was a bit cold. I had breakfast at ~7.00 a.m. and went to the office at ~8.00 a.m. The task for today is to test the system that I upgraded yesterday. Before testing, I had to upgrade my driver first. At first, the system gets stuck when writing the data back to the main memory. I spend time finding bugs from 11.00 a.m. to 6.30 p.m. Eventually, I found the bug. It is simply a single line of wrong condition in the AXI-Stream communication protocol. I had dinner at restaurant 1 and worked until 9.30 p.m. Finally, I headed back to my bedroom.

Thursday, 17 July 2025

I woke up at ~6.30 a.m. The weather was a bit cold. I had breakfast at ~7.00 a.m. and went to the office at ~8.00 a.m. The task for today was to test the fixed system from yesterday. The system testing result was good. I then started to deploy the machine learning. At first, the Vitis program told me that the system was too large. I then reduce the size of its variable, and it looks good. Unfortunately, when testing, the DMA was stuck. It seems that the DMA did not get any result from the model. I get back to my accommodation at ~10.00 p.m.

Friday, 18 July 2025

I woke up at ~6.30 a.m. The weather was a bit good. I had breakfast at ~7.00 a.m. and went to the office at ~10.00 a.m. The task for today is testing the practical machine learning model on my new Magic Streamer infrastructure. The model was like the one in the picture below. The red line is where the model was chopped into multiple pieces. All pieces will be scheduled to reprogram using my chip. However, the testing failed. The third piece did not send the data back to the memory. That was extremely frustrating. Therefore, I have to build some hardware to debug the system to check where it is wrong. Unfortunately, the debugger returns me cranky results. That is, when I bind the last stage to the magic streamer, the magic streamer's debugger shows that the data was sent to the streamer already. In contrast, when the SAME stage is entered into the DMA IP (direct memory access). Its debugger showed that there is no data being sent. Why does the same chip provide different results? That drives me crazy.

Saturday, 19 July 2025

I woke up at 7.30 a.m., the weather was good, and I went to the office at 9.00 a.m. The task for today was to find where the issues are. I started by building a simple hardware design to validate the hardware generated by my HLS4ML, as shown in Figure 32.

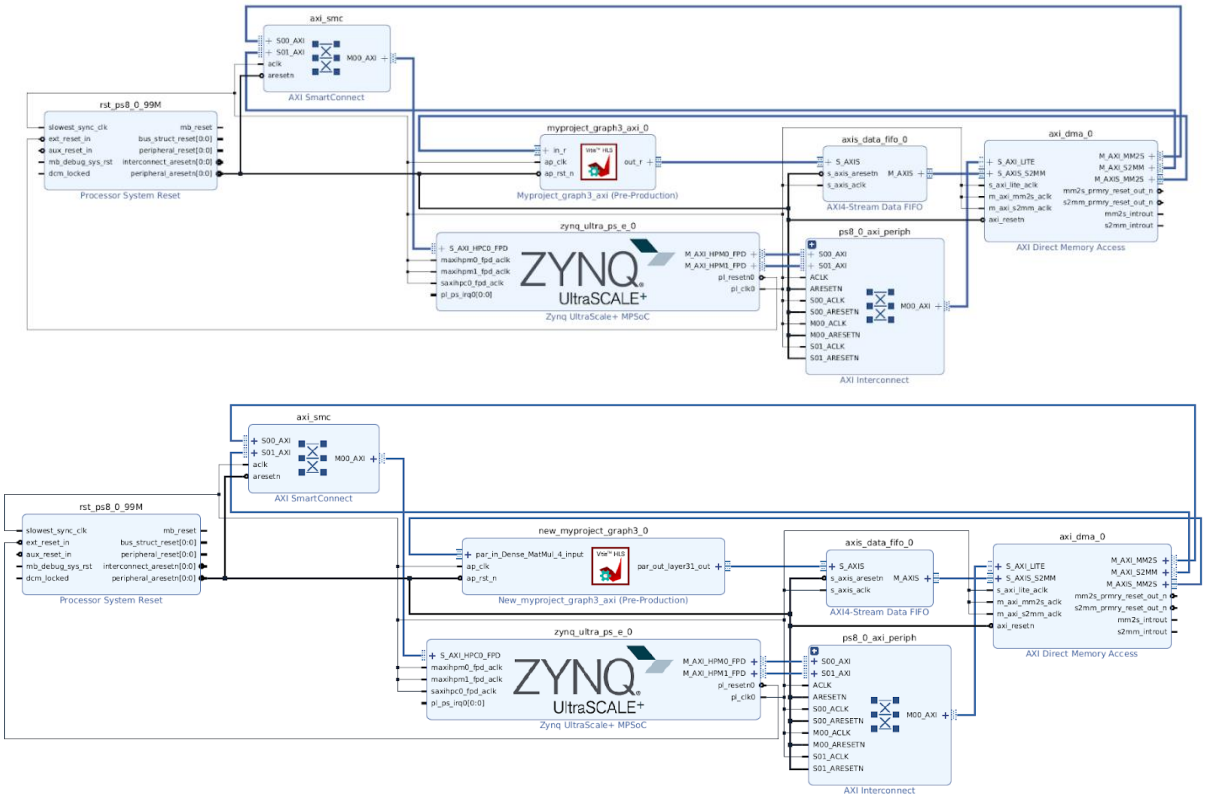


Figure 32 The Design for Validation (without DFX)

The project block in the upper figure is generated by the old HLS4ML backend, but the block generated by the lower figure is generated by my new HLS4ML backend. Fortunately, my new hardware block is stuck. Now we know who the issue is. I continued to dive into the generated code, and I found that it is only a single line of code that makes the data transfer

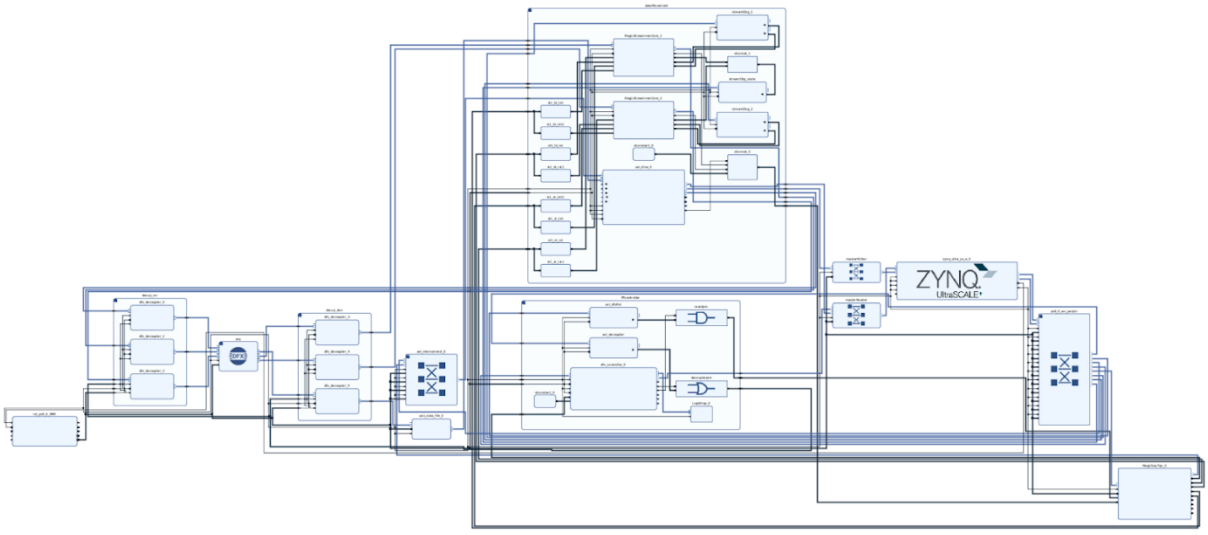


Figure 33 Vivado Block Design for 2 connected Magic Streamer + 1 DMA

incorrect. After having lunch at restaurant 1, I started to redesign the whole hardware block (Figure 33)to make it easy to read and debug. I headed back to restaurant 1 and tested the system there until 10.00 p.m.

Sunday, 20 July 2025

I woke up at ~7.30 a.m., I had breakfast at ~ 8.00 a.m., and headed to my desk. The task for today is to test the whole design that I had rebuilt yesterday. The result is correct, and the magic streamer operates as it should. The result is shown in Figure 34.

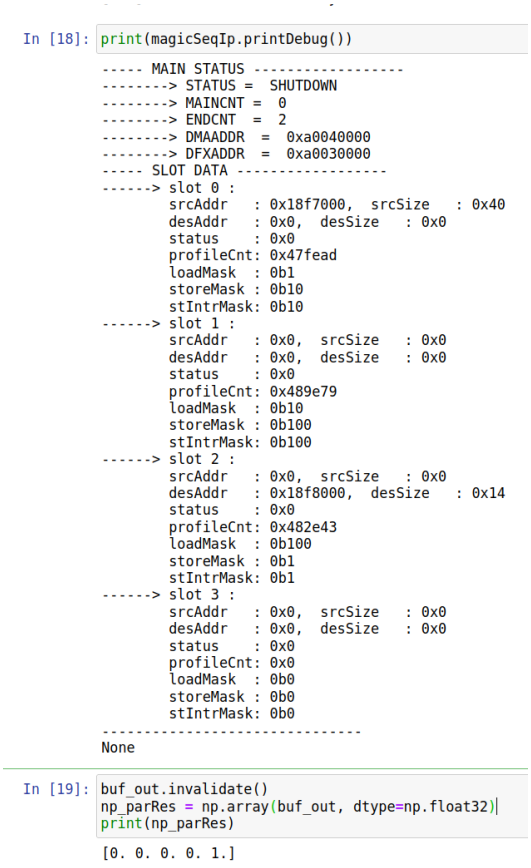


Figure 34 The Result from MagicStreamer Debugger

After validating the task, I started to modify the HLS4ML to make it simulatable. It is essential because designers must use it to validate that the results from the hardware and software are identical. For your information, the HLS4ML simulator will generate the C++ code that represents the machine learning model that will be used in hardware code. I need to add this feature to our backend because my backend must support multiple inputs and outputs that the other backend does not support. At 8.00 p.m., I went to my Hostel reception to recharge my laundry card; however, the reception staff told me that she could only recharge it on Monday. Now I have to wash my clothes by hand.

Monday, 21 July 2025

I woke up at ~8.00 a.m. That was a bit late; it seems that my standard was lower. That was pretty bad. I can't allow myself to lower my bar anymore. I headed to my desk at ~9.30 a.m. The task for today is to complete the simulator generator and test the simulation system. There were many bugs in the generated file that I had to deal with. Fortunately, the test was passed as I expected (Figure 35)

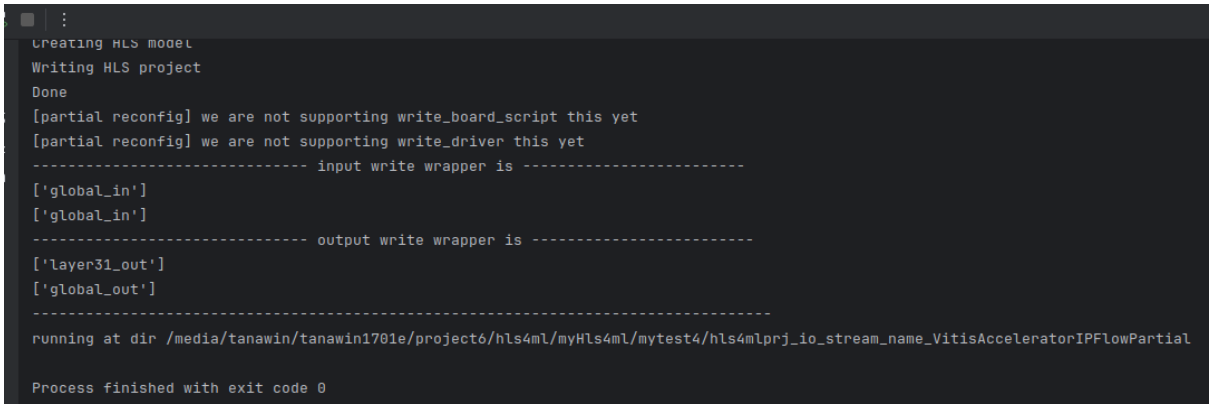


Figure 35 Simulation Success on HLS4ML

However, that is not all for today; it is only the functional simulation of the Machine learning model. I still have to build the simulation for csim/co-simulation. In addition, the co-simulation is a simulator built for hardware simulators. This process takes much longer than a functional simulator because it simulates in detail the hardware implementation. It is essential for the FIFO optimization feature. I headed back to restaurant 1 at 6.30 p.m. and worked there until 9.30 p.m.

Tuesday, 22 July 2025

I woke up at 4.45 a.m. I started cooking breakfast by myself. The menu was instant soup with crackers. I feel a bit sorry for my kidney because it is too salty. While cooking, I let the washer do its work. I headed to my office at ~ 7.30 a.m. The task for today was to test the co-simulator that I designed yesterday. The test was successful. Next, I have to upgrade the simulation testbench to support HLS4ML's multigraph feature. The feature was to chop the machine learning model into multiple pieces as Figure 36.

In our co-simulator, I will chop the DENSE_COMMON so the block will be divided into two pieces. Our new simulator must handle this as well. Fortunately, the simulator passed for testing. I headed back to restaurant 1 and worked there until 9.30 p.m.

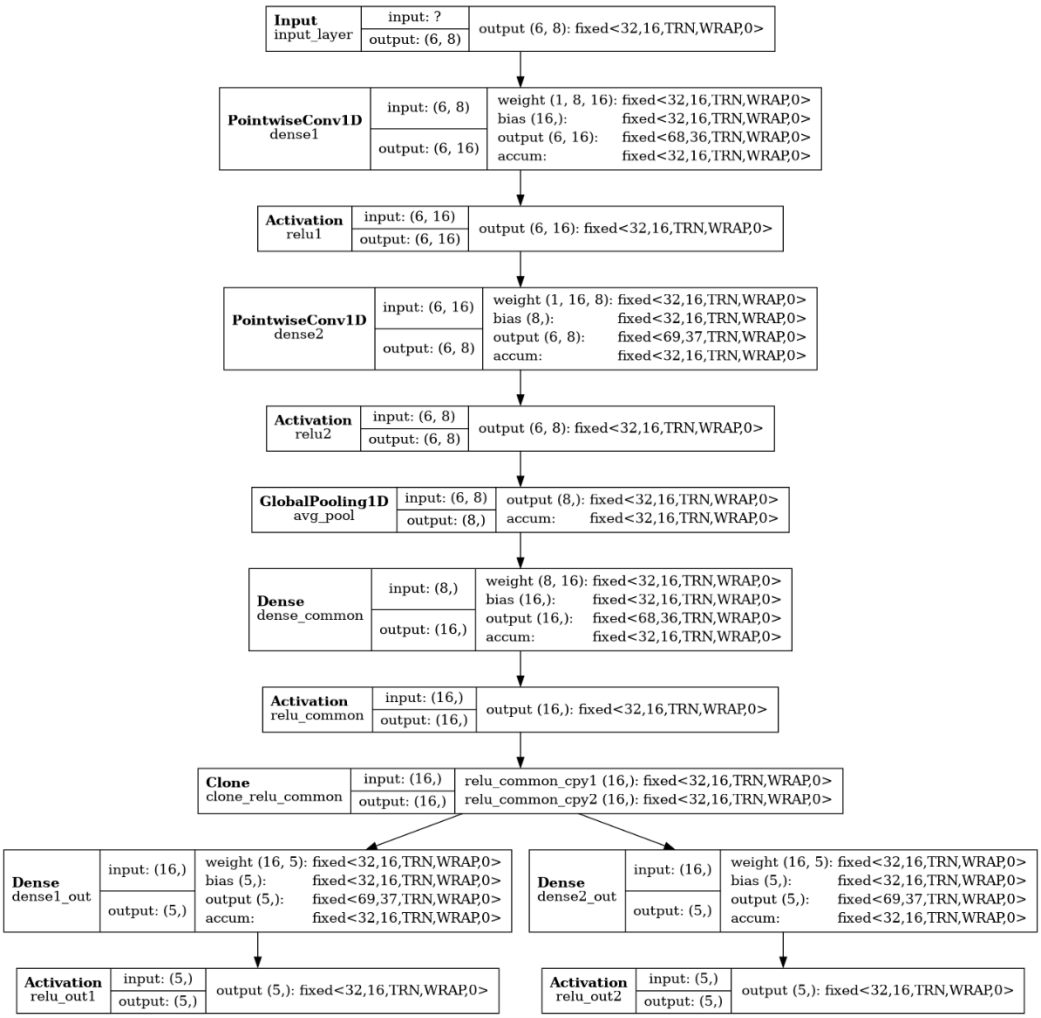


Figure 36 Multiple Output ML Graph

Wednesday, 23 July 2025

I woke up at ~7.30 a.m. The weather was good. I had breakfast at restaurant 1 and then headed directly to my office. The task for today was to create my slide that I have been working on this week and send it to my supervisor. The slide is in this link (https://docs.google.com/presentation/d/1T-h_4ERSRtcp-obVTphe4Y7mLSzVBprWum1-6GDel1o/edit?usp=sharing). It intends to tell about the current architecture, bug report, simulation procedure, simulator backend comparison, and current progress. After that, I started to finalize this report because the THAI agency reminded me to send it. After having lunch, I started to redesign the magic sequencer, the partial reconfiguration controller, to support an

interrupt signal. It will be not only useful when we are trying to profile the system, but also resourceful when the system is not polling the command to check the status of the system.

Figure 37 was the upgrade register architecture for supporting the interrupt system. The interrupt register will be set to 1 when all partial regions have been deployed and executed sequentially.

At 6.30 p.m. I headed back to restaurant 1 for dinner and worked there until 9.40 p.m. After that, I headed back to my accommodation.

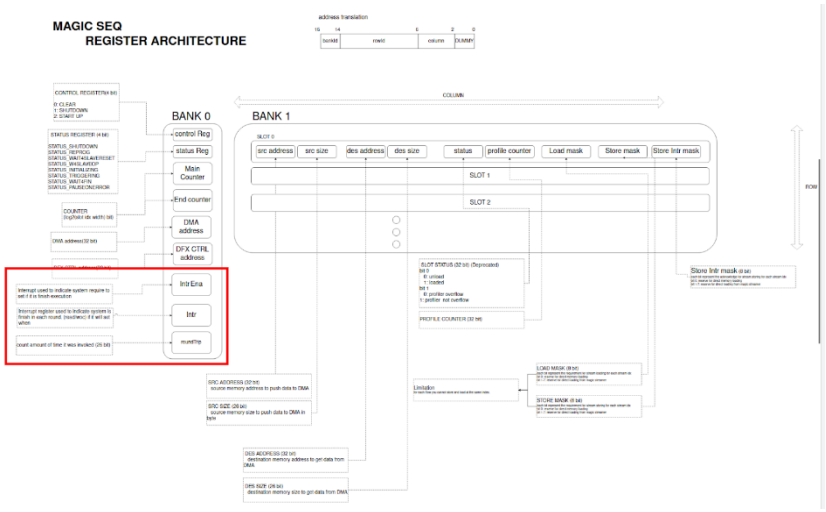


Figure 37 Upgraded Magic Sequencer IP

Thursday, 24 July 2025

I woke up at 7.30 a.m. The weather was a bit cool, so I headed to the office at 08.30 a.m. The task for today was to continue upgrading the Magic sequencer from yesterday. However, for today, I did not fully utilize its feature about the direct start/interrupt signal; I directly connected the new system to the interrupt module as shown in Figure 38. This mechanism allows the MagicSequencer system to trigger the central processing unit and notify of the successful execution status.

Unfortunately, the first time synthesis failed due to large resource usage on the chip. I fixed it by reducing the Machine learning network layer and quantizing (reducing the precision of the floating numbers) the value. At 6.30 p.m., I headed back to restaurant 1 and worked there until 9.40 p.m. The evening task was to debug the system that falsifies the output compared to the software simulation. When I got back to my accommodation at 10.00 p.m., my summer student friend bought me lemon tea. I am very appreciative of that.

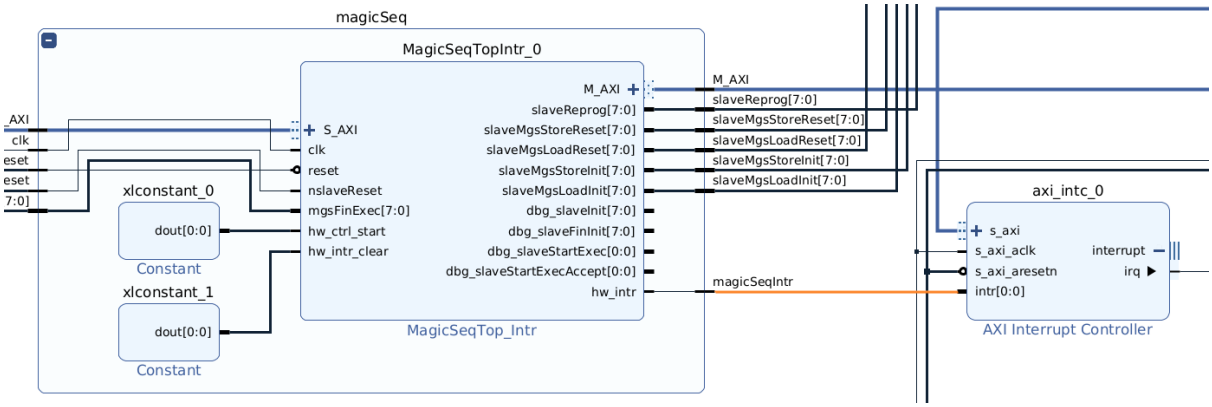


Figure 38 Interrupt Design for Magic Sequencer

Friday, 25 July 2025

I woke up at ~6.45 a.m. The weather was a bit cool, so I continued to inspect which system falsifies the output at ~8.17 a.m. Eventually, I saw that the generated code missed some temporary variables. That makes the system retrieve the data from the same FIFO (first in, first out). When it retrieves from the same buffer, it causes the FIFO to empty more quickly than we expected; therefore, there is no data out anymore. After the problems were resolved, I started to stress test my system by feeding 1000 queries through the new machine learning hardware model. Unfortunately, the result showed that the time used to reconfigure the hardware produces too much overhead compared to the inference time (machine learning execution time). In the evening, I headed back to restaurant 1 for dinner and worked there until 10.00 p.m.

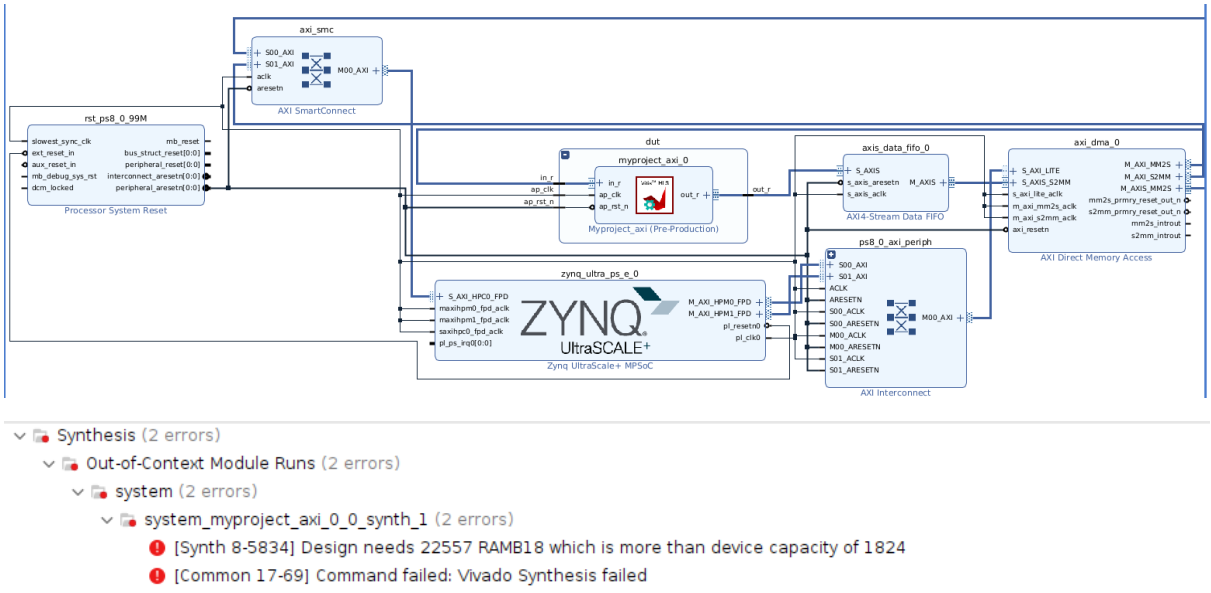


Figure 39 Error on Vivado for U-Net Synthesis

Saturday, 26 July 2025

I woke up at ~7.45 a.m., I had breakfast at restaurant 1, and headed to my office at ~9.00 a.m. The task for today was to test the big U-Net model. Actually, I left the computer to synthesize the U-Net model all night; however, when I tried to compose the U-Net hardware into the whole simple hardware data structure in the Vivado program. The program showed that the design is too huge and cannot fit into the FPGA fabric. The design errors are shown in Figure 39.

I gave up on this first; I changed my mind to focus on the next topic. It was the overhead between the configuration time and the execution time. I started to modify the hardware to support the experiment until 8.00 p.m. Then, I went running around the CERN. I began running earlier than before, and the environment was a bit lively. This is the first time that I have seen a cat in Switzerland. Actually, it is my first animal that I saw in Switzerland.

Figure 40 The cat that I saw while exercising

Sunday, 27 July 2025

I woke up at ~7.00 a.m. The weather was pretty cold, so I headed to my office at 8.17 a.m. I started to experiment with the design that I created yesterday. The experiment tends to observe the overhead from the reconfiguring time on the FPGA compared to the execution time. In the ideal goal, we need to minimize the reconfiguring time and keep the FPGA executing the machine learning workload. The result shown in the picture below indicates that the system produces less overhead when queries are higher.

At ~3.00 p.m., I start to create the presentation slide for Tuesday's meeting. My slide focuses on three main parts as follows: The current architecture, the specification of the new HLS4ML backend, and the result of the experiment that I did today.

(https://docs.google.com/presentation/d/1F5tK0TS-VKrrMX7-Purq7FiEsPUIuqp991WBt-E7Ko0/edit?usp=sharing)

Monday, 28 July 2025

I woke up at ~7.30 a.m. The weather was pretty cold, so I headed to my office at ~9.00 a.m. The task for today was to upgrade the backend, which is used to convert the machine learning model into synthesizable hardware language. The upgrade's purpose is to migrate from using Vitis_hls as a backbone to Vitis Unified. We have to leave Vitis_hls because it will be deprecated by AMD in a few releases, and it is lacking on my machine for unknown reasons. At ~6.30 p.m., I went back to restaurant one and tried to figure out how to pass the C++ declaration flag through the new Vitis Unified backend until 9.30 p.m.

Tuesday, 29 July 2025

I woke up at ~7.30 a.m. The weather was extremely cold, at ~8.45 a.m. They started to recheck the presentation slide (https://docs.google.com/presentation/d/1F5tK0TS-VKrrMX7-Purq7FiEsPUIuqp991WBt-E7Ko0/edit?usp=sharing) until 9.45 a.m. I went to building 40 for a group meeting with my colleagues and my supervisor. We focus on what work we are doing and discuss the issue with my system. After the meeting, I went to a personal meeting with my supervisor at building 41, focusing on the task that I will do this week. In the afternoon, I started working on the task by converting the interface of the machine learning to directly match the magic streamer. I headed back to the restaurant 1 at 6.30 p.m. and worked there until 9.30 p.m. In addition, the new interface that I am trying to convert aims to reduce the memory space usage. For example, the raw data produces 4 bits, and the old interface will convert it to 32 bits, which is the standard value for a computer system. In contrast, to utilize the hardware resource on custom Hardware, we don't have to comply with the standard system. I get to sleep at ~10.30 p.m.

Wednesday, 30 July 2025

I woke up at 5.00 a.m. The weather was extremely cold, so I started cooking at restaurant 1. There is no one there. I headed to my office at ~7.00 a.m. The task for today is to make HLS4ML, a machine learning hardware model code generation, which can generate code like yesterday automatically. After it can be generated, the preliminary result shows that it requires too much FIFO for each layer, as shown in Figure 41. It is a normal situation that can occur for the first time. To mitigate this problem, we have to use a FIFO optimization technique. This

technique uses the chip simulator to simulate the observed FIFO usage in each layer. We use the observed data to update the actual hardware design.

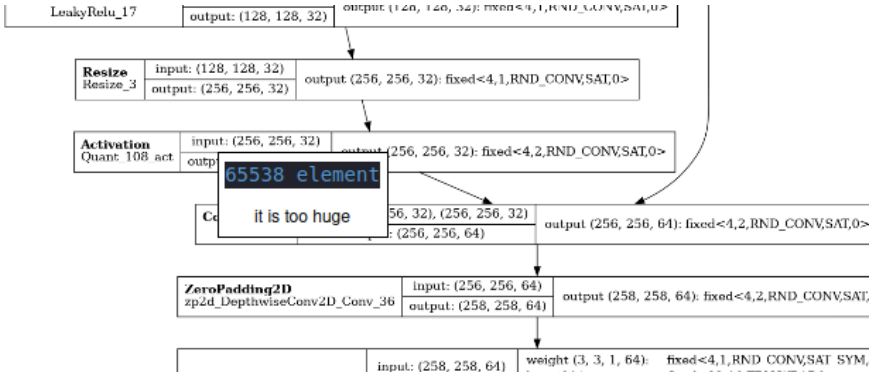
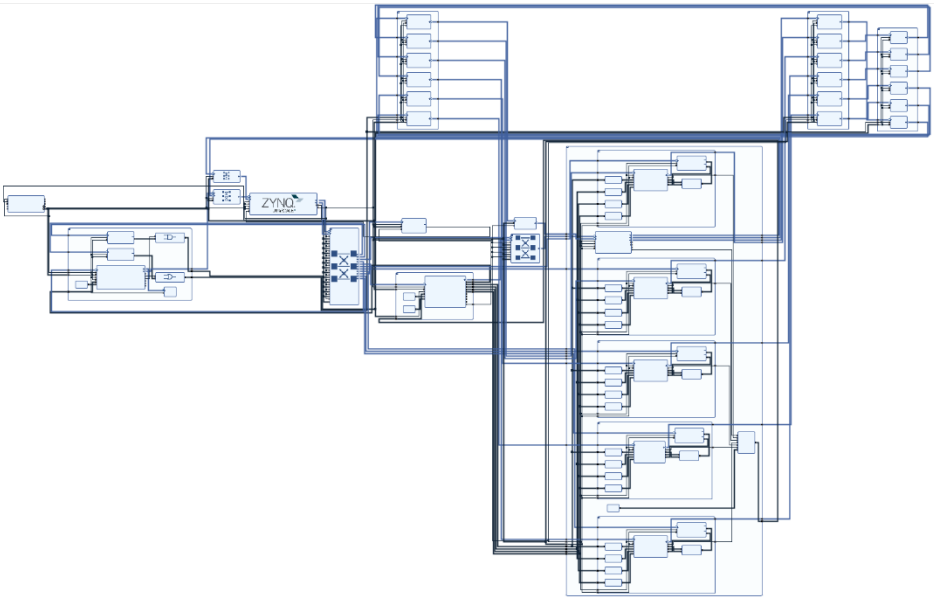


Figure 41 Buffer Usage Visualization for U-Net

I headed back to restaurant 1 and worked there until 9.30 p.m.

Thursday, 31 July 2025

I woke up at 7.00 a.m. The weather was pretty cold. I headed to my office at ~8.30 a.m. The task for today was to continue the FIFO optimization from yesterday. The procedure took more than 5 hours to simulate. In the meantime, we started to upgrade the hardware infrastructure used to support this model. We have to upgrade the infrastructure because the new model is much larger than the previous model; therefore, we have to update the model to support larger channels and memory to hold the data from the new configuration module. The new picture is like Figure 42.



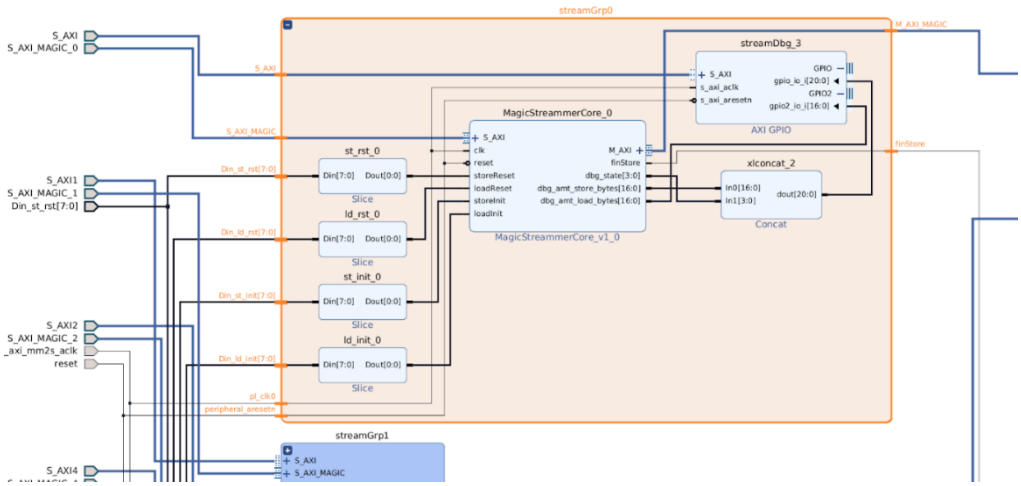


Figure 42 Magic Stream Grouper IP

At 6.30 p.m. I headed back to restaurant 1 and worked until 9.30 a.m. to figure out the port generation in the HLS4ML module.

Friday, 1 August 2025

The weather was pretty good. I headed to my office. The problem from yesterday was that the port generation was twice as big as the specification for two times. For example, the specification says that the port should have 4 bits as an output, but the actual generated port provides 8 bits as an output. I tried many ways to induce the Vitis to generate the desired port size; however, the Vitis did not comply. Eventually, I decided to build another hardware design block to deal with size conversion instead, as shown in Figure 43.

After synthesizing the system, I have found that the system overutilizes the result usage. I considered many optimization techniques, but they did not work.

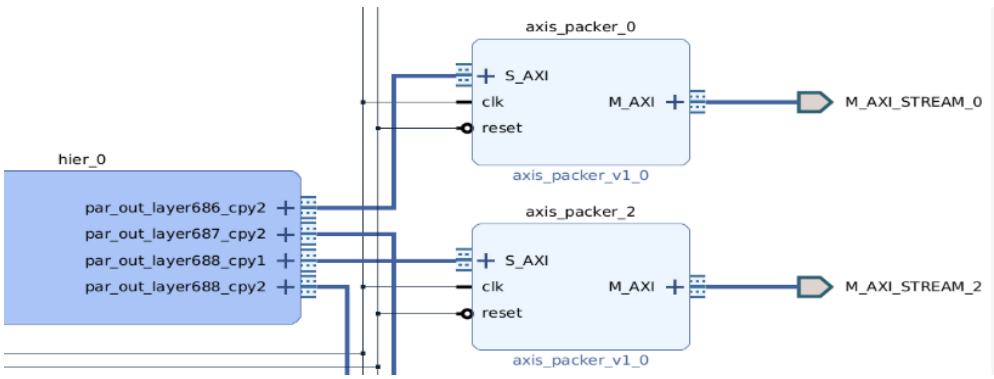


Figure 43 AXI Data Packer

Saturday, 2 August 2025

The weather was pretty good. After having breakfast at restaurant 1. I headed to my office at ~ 9.00 a.m. My goal for today was to fit our HLS4ML model into the FPGA. It was the first time that I tried to do strange floor planning for a partial reconfiguration module.

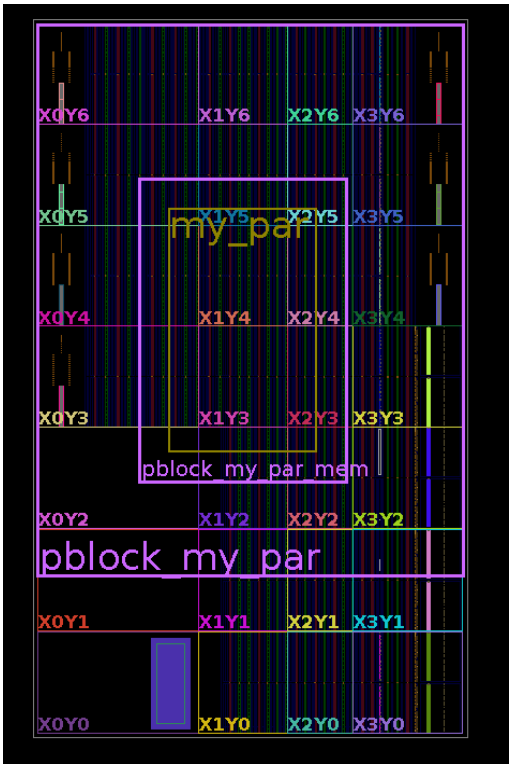


Figure 44 pblock_my_par Floor Planning in Vivado

As depicted in the Figure 44, the pblock_my_par was supposed to be the reconfiguration module for the HLS4ML machine learning module for all resource types except memory blocks. In contrast, the pblock_my_par_mem was supposed to reserve the memory for the machine learning module. Unfortunately, I failed; the Vivado program says that it cannot route the design.

Sunday, 3 August 2025

I woke up at ~7.30 a.m. I headed to restaurant 1 at ~9.00 a.m. I started to mitigate the machine learning model deployment by chopping the machine learning into three parts, as shown in Figure 45.

Unfortunately, I discovered the bug from HLS4ML's backend generation; the port at the last chopped section has the conflict name. After fixing the bug, I still encountered the same issue about resource overutilization.

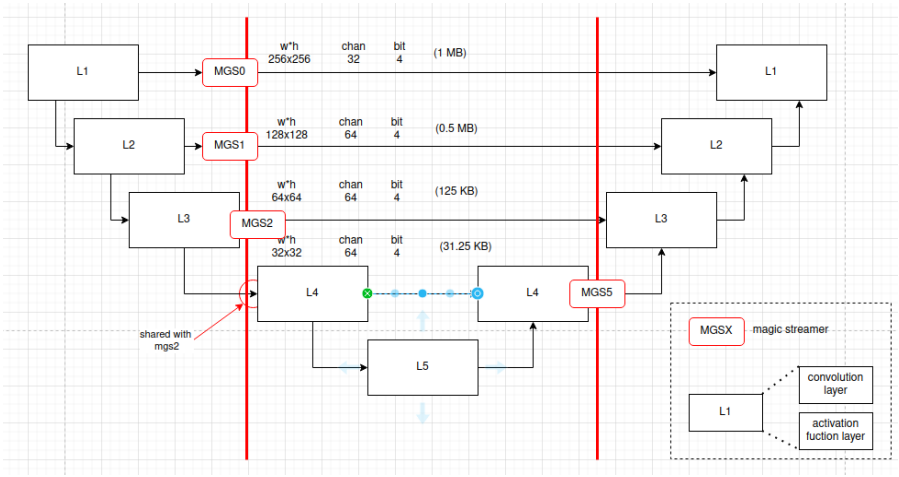


Figure 45 Seperating Point on U-Net

Monday, 4 August 2025

I woke up at ~7.00 a.m. After having breakfast at restaurant 1, I headed to my office at building 653. I chopped the Machine learning model into 4 parts, hoping for resource fitting. Moreover, according to the result from yesterday, the system requires the lookup table(LUT) resource for the dynamic region. But the static region requires more memory for the magic streamer, as shown Figure 46. I spend a lot of time trying to reconfigure the constraint to fit the resource. Eventually, I can fit the two sections (L1 -> L3, L4 -> L5.5) into the FPGA, but the last two sections still cannot fit into the FPGA. The resource usage described in Figure 47.

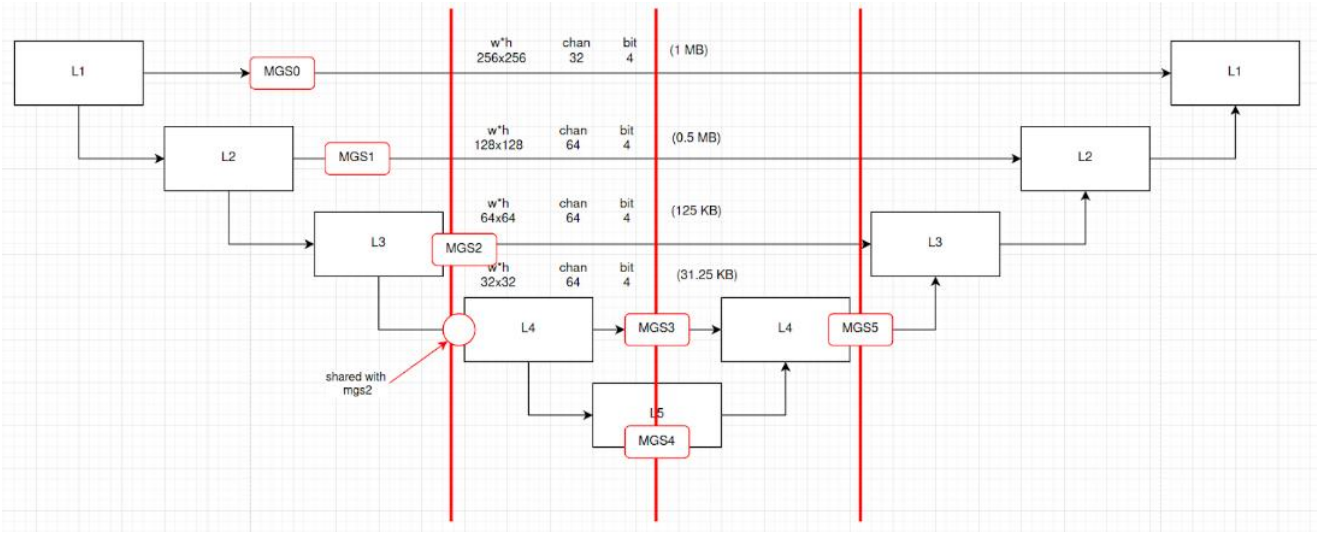


Figure 46 New Separating Point on U-Net

OLD	lut	ff	bram	dsp	
Unet-1	122678	29000	10	288	
Unet-2	155242	35991	0	288	
Unet-3	225399	64677	0	288	too huge
Unet-4	230261	81667	0	256	too huge

Figure 47 Resource Usage Report for 4 U-Net Subgraphs

Tuesday, 5 August 2025

I woke up at ~6.00 a.m. I have to create a quick presentation slide for today's meeting with my supervisor and lab colleague. The presentation slide is at this link (https://docs.google.com/presentation/d/14w3Ydswqq6FwoOLzY5VtpzuIXF4GnGBao8IP59XkkBw/edit?usp=sharing). The slide mainly describes the magic sequencer (buffer served to hold the data in the FPGA fabric) saving, how the ML graph was chopped, and the new partial reconfiguration partitioning. The interesting part is that I can allow some bram blocks, supposed to be static blocks, to invade the dynamic region, as shown in Figure 48.

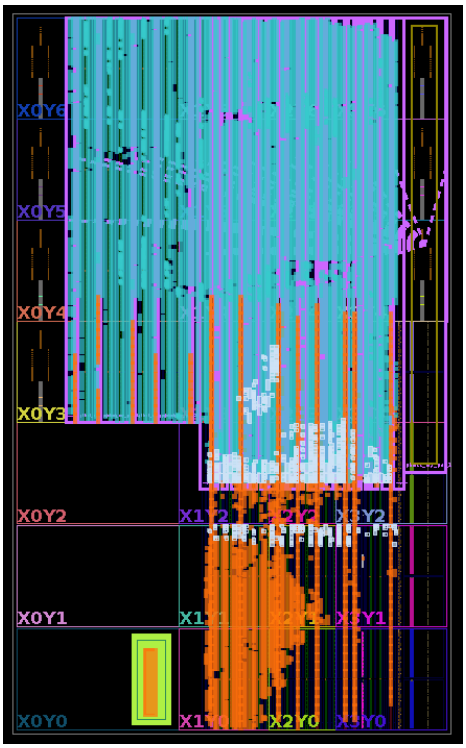


Figure 48 New Floor Planning Design

After meeting with the supervisor, I had to prepare another slide which will be used for an oral presentation on 7 August.

Wednesday, 6 August 2025

I woke up at ~7.00 a.m. I attended my summer student friend's oral presentation from 9.00 a.m. to 12.00 p.m. Many of them talk about physics; I don't know what they have worked on or talked about; however, I am sure that they are having a good session in their field. After the session, I realized that I should better upgrade my slide; therefore, I upgraded my slide all afternoon and headed back to restaurant 1 to practice the oral presentation. (the presentation slide https://docs.google.com/presentation/d/1iPwhcLKuH8etbSyN-wa9xqC40KbNu-vwVE09RptmARY/edit?usp=sharing)

Thursday, 7 August 2025

I woke up at ~6.00 a.m. I started to practice oral presentations for the final time at restaurant 1. I attended a summer student oral presentation at 9.00 a.m. My presentation started at 10.15 a.m. I am a bit nervous because I am afraid I will use it over time, even though I timed myself yesterday already. After the session, I went back to my office at ~1.00 p.m. to continue my work. The task for today is to optimize the logic of UNET. One technique that I use is to increase the reuse factor. The reuse factor technique will reuse mathematical operations such as multipliers. Moreover, I have to modify my new backend to support new optimized IO for code vitis code generation and co-simulation generation. At 8.30 p.m., I started exercising by running around CERN.

Friday, 8 August 2025

I woke up at ~7.00 a.m. I headed to my office to build the UNET model by setting the reuse factor to 8 at maximum. This session takes more than 8 hours to execute. Unfortunately, the resource usage consumes more than the baseline, as shown in Figure 49.

OLD	lut	ff	bram	dsp
Unet-1	122678	29000	10	288
Unet-2	155242	35991	0	288
Unet-3	225399	64677	0	288
Unet-4	230261	81667	0	256
NEW				
Unet-1	131476	63163	0	1839
Unet-2	1412142	67580	0	2520
Unet-3	237293	120536	0	2520
Unet-4	286674	152356	0	2520

Figure 49 Resource Usage Using Different Re-use Factor (1 OLD)/(8 New)

I tried to debug it all day to find where it was wrong. I attempted to increase the reuse factor from 8 to the maximum possible. When synthesizing the hardware, the system gets stuck. I discussed with my supervisor that I should test the HLS4ML layer by layer.

Saturday, 9 August 2025

I woke up at ~7.00 a.m. I had breakfast at a restaurant, and I went to my office at ~8.30 a.m. The task for today was to determine which position would slow down the system. I started by suspending all Neural network layers except the single Depthwise layer. After that, I started to break down the related code to see how the Depthwise implementation works. The concept of Depthwise is identical to software programming; however, the implementation is a bit different. My assumption is that the synthesizer does pipelining while the system is multiplexing. That makes the system generate unnecessary logic. Therefore, I try to confirm my assumption by disabling the pipeline, and it works! Unfortunately, disabling the pipeline makes the system sacrifice its performance. Eventually, I went back to my accommodation at ~10.00 p.m.

Sunday, 10 August 2025

I woke up at ~8.00 a.m. The weather was good. After having breakfast at restaurant 1, I headed to my office at building 653. The resource usage from yesterday was good; however, the timing performance was unacceptable. Fortunately, my supervisor gave me advice to insert the initialization interval in the main Depthwise loop system. The result was pretty good. After that, I tried to find many configurations to maximize the system design resource usage and runtime performance.

Monday, 11 August 2025

I woke up at 7.00 a.m. The weather was good. After having breakfast at restaurant 1, I went to my office at 8.45 a.m. The task for today is to regenerate the testing project from yesterday because we observed that the product kernel must be changed when the reuse factor exceeds the N_OUT size. After experimenting with the performance, it was discarded and described yesterday's performance.

That disappoints me a bit. After we had the correct performance evaluation, we can conclude that the ReuseFactor for Depthwise convolution and Pointwise convolution should be set to 4 and 8, respectively. I started to synthesize the whole UNET again. For U-Net, the synthesis result is going well, but when I run the implementation, the last graph utilization shows

a weird result. In the evening, I headed back to restaurant 1 to debug it and prepare a presentation slide for tomorrow until 10 p.m.

Tuesday, 12 August 2025

I woke up at 8.09 a.m. The weather was a bit hot. Today, I woke up late, and the presentation had not finished yet. I quickly created the presentation slide about the depthwise bug and the solution that I proposed. Then, I went to a meeting at 10.00 a.m. This session was good, I got a lot of feedback from my supervisor and co-supervisor. The discussion for today was very interesting. After the meeting, I then met in person with my supervisor to discuss the task for this week. We agree that we should start building the tutorial for the HLS4ML system. In the afternoon, I go to the CERN gift shop and get back to my office in building 653 to start creating the demonstration task.

Wednesday, 13 August 2025

I woke up at 5.30 a.m. The weather was 19 degrees Celsius. After having breakfast at restaurant 1, I headed to the office at 7.30 a.m. The task for today was to test the system before building the HLS4ML tutorial. The test model was a simple UNET that we used in the last two weeks. However, it must be tested with the new port generation that I mentioned last week, at 12.00 p.m. Unfortunately, the results from hardware and software are not identical. So. That is very weird. Therefore, I create an assumption of an error point and test for each of them to identify the error point. Unfortunately, all assumptions are eliminated. In the evening, I resume my work by creating all of them again from the beginning. Luckily, the system provides the correct result. In addition, I guess the system provides an error because the hardware infrastructure clock frequency is missing when gathering by Vivado.

Thursday, 14 August 2025

The task for today, after fixing the bug, is to build the Vitis Unified Backend. This backend does not support the chopped machine learning graph that I got stuck on yesterday. However, I have to build it because we should have a baseline to compare with it. I start building it by mimicking other backends. It is a bit of a big task because the backend is supposed to generate the whole related hardware and kernel infrastructure. In addition, I introduce the V++ platform linking approach. This approach will automatically combine the HLS4ML kernel with the infrastructure seamlessly without requiring a manual TCL script for infrastructure creation.

Friday, 15 August 2025

The task for today is to continue the task from yesterday. I focus on building the wrapper that connects to the platform. The wrapper of the system is responsible for managing the AXI memory map port to retrieve the data from main memory, convert it to the HLS4ML array, and push it to the input stream. The wrapper also converts the data from the HLS4ML array and writes it back into the main memory.

Saturday, 16 August 2025

The task for today is to test the wrapper built yesterday. In the current session, we use Vitis Unified IDE, a graphical user interface program, to make the HLS4ML kernel project and create the system project. The system project will link the platform and the kernel together and provide the bitstream file as an output. After that, I test it on the actual hardware, and the result looks good.

It raises a question. How can I make it work automatically using the command line instead of doing it manually using the GUI? I am first trying to use Python vitis-cli to interact with the Vitis subsystem. Unfortunately, Xilinx doesn't provide sufficient documentation and tutorials for my workflow. I then head back to the traditional approach, which is using a batch script to invoke the Vitis system directly.

Sunday, 17 August 2025

The task for today is to continue integrating the automation script into HLS4ML. The system can automatically synthesize and build the block design. The block design is shown in Figure 50.

I then start comparing the result from the bridge simulator and the PYNQ hardware (ZCU102), and the result works fine. I then continue to integrate the PYNQ driver generator that suits the platform into the HLS4ML framework.

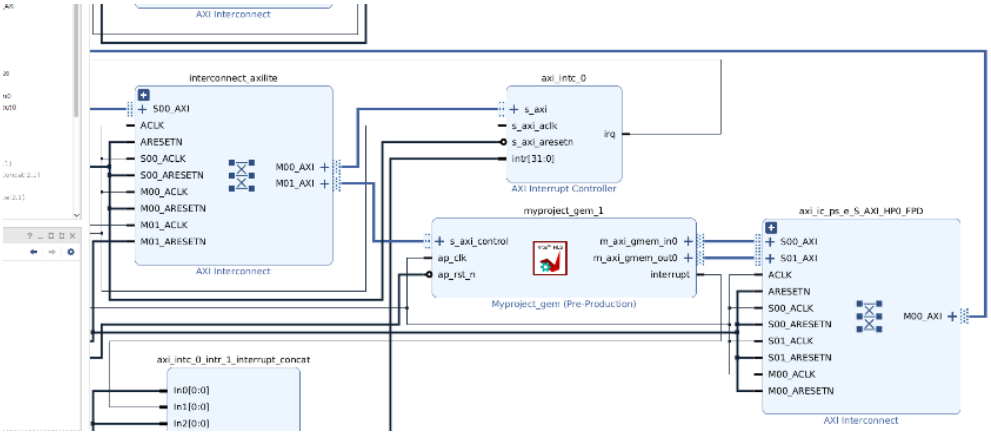


Figure 50 The Block Design Generated by V++

Monday, 18 August 2025

There is one day left before the group presentation, so I will use today to explore the possibility of integrating the partial reconfigurable region in the Xilinx platform project. My investigation has hit a roadblock. The v++ linker used to link the platform and HLS kernel only supports the AXI memory map linker. Unfortunately, the HLS4ML io port that we used with Magic Streamer and Magic Architecture is AXI stream, which is not supported by the linker.

Tuesday, 19 August 2025

I woke up at ~ 8.00 a.m. The weather was good. After having breakfast at restaurant 1, I immediately created a quick presentation slide for today's meeting at 10.00 a.m. The meeting is great. After the presentation, my supervisor told me that he successfully reserved the presentation time slot for my project for the HLS4ML community meeting this Friday at 5.00 p.m. In the afternoon, I then fixed the issue from yesterday by using a traditional TCL script to auto-generate the hardware block design and connect the HLS4ML kernel.

Wednesday, 20 August 2025

The task for today is to continue building the automation script. I started creating the Verilog generator to aggregate the various sizes of magic streamers together into a single IP called the streamer group. Moreover, in HLS4ML, we still need a separate script package for this IP at runtime. We cannot statically prebuild the magic streamer group because the model that

users may cause various magic streamer configurations. One example of a magic streamer group is shown in Figure 51.

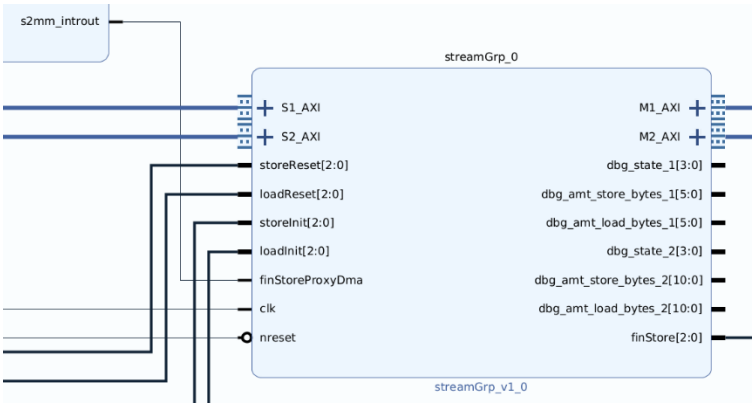


Figure 51 Stream Group IP

Thursday, 21 August 2025

I continue to build the automation script from yesterday. After we have the stream group IP, we have to integrate it into the whole system infrastructure, as shown in Figure 52.

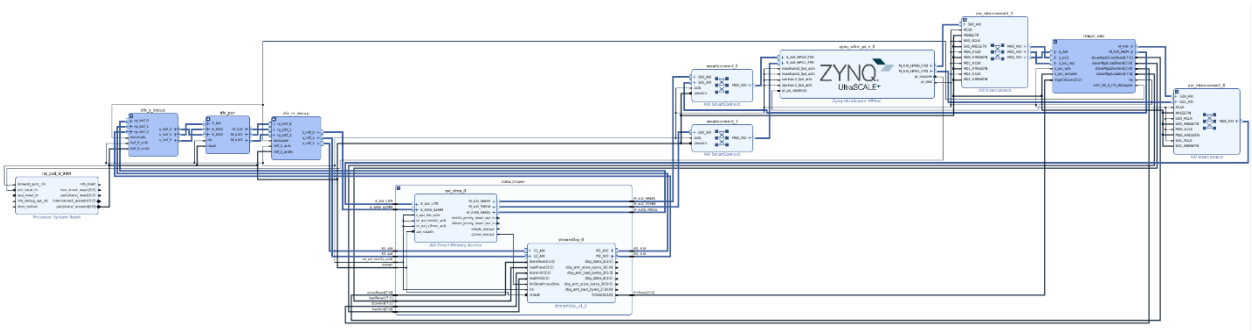


Figure 52 Entire System Written for Magic Architecture Support

Writing the script from scratch is too complicated. I then use the Vivado GUI to build the block design and export it as a TCL script. I then broke that code down and built my own version.

We cannot use the script generated from the GUI directly because we have to dynamically add the HLS4ML kernel at runtime.

Friday, 22 August 2025

The task for today is to prepare a presentation slide of my work at the HLS4ML community for today, 5.30 p.m. The slide is at this link https://docs.google.com/presentation/d/1gRxVzgEY_64keN2yhuQTR3etD4acOYJCc8EIpzW18ic/edit?usp=sharing. Thankfully, my supervisor gave me advice and comments for my slide.

Saturday, 23 August 2025

I woke up at ~ 8.00 a.m. The weather was good. After having breakfast at restaurant 1, I take tram T18 to meet my senior who came for the CERN teacher training program. We go to St Pierre Cathedral as shown in Figure 53, and then go to the broken chair placed across the street from the Palace of Nations. We then have lunch at a Thai restaurant at Geneva airport and go to see her off at the airport. After coming back to CERN, I resumed my work. In addition, the task for today is to create the TCL script that creates the subsystem for HLS4ML dynamic partial reconfiguration. The auto-generated feature is shown in Figure 54.

Figure 53 St Pierre Cathedral

However, this system lacks machine learning model integration. It is just the infrastructure of the hardware. Finally, I get back to my accommodation at ~ 10.00 p.m.

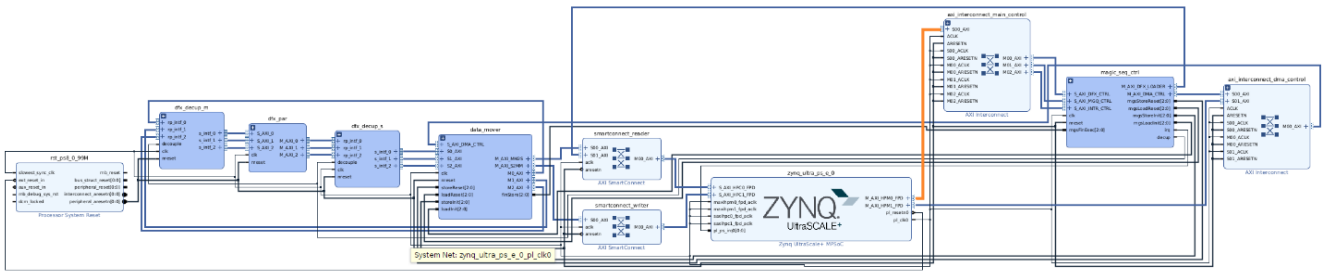


Figure 54 Entire System Generated for Magic Architecture Support

Sunday, 24 August 2025

I woke up at ~ 7.30 a.m. The weather was pretty cold. After having breakfast at restaurant 1, I head to my office at building 653. The task for today is to explore how to match the machine learning kernel port generated from HLS4ML into the structure built yesterday. Unfortunately, since HLS4ML is written in Python, HLS utilizes dynamic type and its attribute declaration. It means I can track the system by simply reading the code structure. That makes the progress slow down.

Monday, 25 August 2025

I woke up at 5.30 a.m. The weather was a bit cold. Today, I cooked breakfast and packed lunch. I head to my office at building 653 ~7.00 a.m. The task for today is to build the MGS controller builder for HLS4ML Vitis Unified Partial Backend. When the machine learning model is chopped into multiple parts, there are input and output ports that must be connected to the valid magic streamer (the intermediate buffer) in the valid order. Therefore, I designed the controller system as shown in Figure 55.

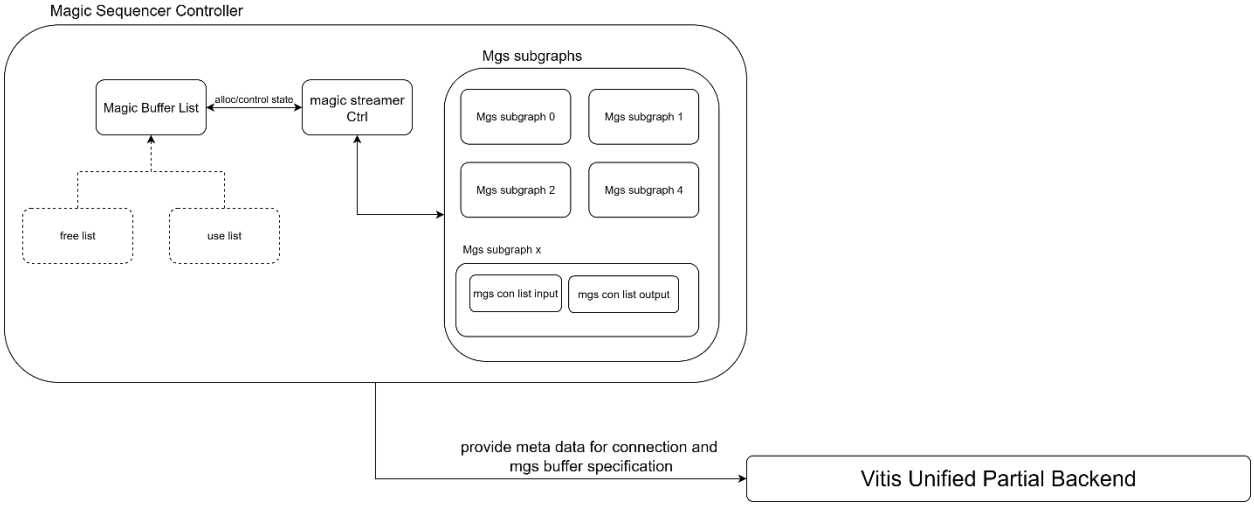


Figure 55 Magic Streamer Manager on HLS4ML

Next, after the model is identified using the system above, the HLS4ML is supposed to integrate the HLS4ML kernel into the FPGA subsystem. In this case, we use ZCU102; therefore, I have to integrate the Vivado TCL script to automatically build the FPGA system.

Building the script is quite difficult because the FPGA subsystem has Brobdingnagian subsystems such as interconnect, DMA, magic streamer, magic streamer controller, and region. I work there until 0.00 a.m.

Tuesday, 26 August 2025

I woke up at ~ 8.30 a.m. I have breakfast at restaurant 1. Before meeting with my supervisor today to update my current progress, I have to create the update slide for the presentation. The slide was at this link https://docs.google.com/presentation/d/1WDid5pljWQmJtzJpoflimXPm5rOZFGd8gybl2tvF95c/edit?usp=sharing. After meeting with my supervisor, I got the tasks for this week. The task is to complete the Vitis unified backend. In the afternoon, I then started drafting the missing essential feature. That is bridge simulation, co-simulation, and FIFO depth optimization.

In practice, these features should not be independently coded because we build both the Vitis Unified Backend and the Vitis Unified Partial Backend. Both backends should share as much code as possible due to maintainability.

In the Afternoon, I therefore start refactoring the code before adding these features. At 9.00 p.m., I get back to my accommodation.

Wednesday, 27 August 2025

I woke up at ~ 8.00 a.m. I went to my office ~ 9.00 a.m. The task for today is to test the co-simulation feature that I had drafted yesterday. The co-simulation is the feature that simulates the hardware model at the register transfer level. Unfortunately, the machine learning model's prediction result is mismatched with the bridge simulation. In addition, I am quite sure that the bridge simulation provides the correct result. So, we can conclude that there is something wrong with the co-simulation feature.

Fortunately, in the afternoon, I found the bug; it seems that the input/output port buffer depth size does not match the co-simulation size of the testbench. Actually, this problem is pretty complicated to detect; Xilinx never mentions the specification of this parameter. I have to find the clue from the discussion on the internet.

We do not have to relax with it much. As soon as I fix this bug, I will start to test the FIFO-depth optimization feature for the backend. This feature is much easier than the co-simulation feature. Finally, I get back to my accommodation at ~9.30 p.m.

Thursday, 28 August 2025

I woke up at ~ 7.30 a.m. I went to my office ~ 8.30 a.m. The task for today is to test the backend with the actual hardware. For a single query, the system provides the correct output. Unfortunately, the system gets stuck when we send 10,000 queries in each batch. Not only does

the system get stuck, but I also get stuck. I spent more than 3 hours figuring out what was wrong with my model. Fortunately, I have found the problem; my model, which invokes the neural network layer, was fixed only once. At first thought, I think that it was dataflow, it will execute whatever size on the inputs.

Next, I start to write the pytest for the bridge, co-simulation, FIFO-depth optimization feature, and hardware generation feature. I do this because it is essential for other developers to reproduce this procedure when pushing these codes into the shared repository.

Friday, 29 August 2025

I woke up at ~ 7.30 a.m. The task for today is to finalize the auto driver generator for PYNQ. It is supposed to be integrated with the hardware generation feature. I upgraded the driver to support the interrupt system. Actually, some of the code was reused from my work from the first month that I was working on this project. At 5.30 p.m., my supervisor helps me bring the stuff back from my office and do the last update progress meeting until ~7.00 p.m. The last task is to clean the code for making a pull request to the HLS4ML's main branch.
After that, I do a pre-commit installation to check the coding style of my new backend until 6.30 p.m.

Saturday, 30 August 2025

I woke up at ~ 9.00 a.m. After having breakfast in the kitchen, I went to Geneva Cornavin train station to buy some souvenirs to take back to Thailand. Actually, most of them are chocolate. I am a bit afraid of melting during the journey back. In the afternoon, I start packing my luggage. I feel a bit sad about going back to Thailand while packing. After having dinner in the kitchen, I started cleaning my code at the restaurant 1 until 9.30 p.m. After that, I just sat and did nothing at Building 40 until 11:00 p.m.

Sunday, 31 August 2025

I woke up and started packing my luggage again. After checking out from the CERN Hostel, I went to the CERN souvenir shop to buy some souvenirs to bring back to Thailand. All of them are CERN keychains. At 1.27 p.m., I take the tram and bus to Geneva Airport.

Biography

Tanawin Devaveja

Email: Tanawin1701d@outlook.com
Tel: (+66)89-961-4932, (+41)76-515-1701
GitHub: https://github.com/Tanawin1701d

Education

2023 – Present
Chulalongkorn University — Bangkok, Thailand: Master of Computer Science in Computer Engineering, GPA: 3.94

Thesis -> Kathryn: The Hybrid approach for Hardware design, simulation, generation, and profiling
Thesis extension -> Carolyne: Exploring a Reconfigurable Out-of-Order SuperScala Processor with the Kathryn Framework

2019 – 2023
Chulalongkorn University — Bangkok, Thailand: Bachelor of Engineering in Computer Engineering, GPA: 3.59

Graduated: 2023

Experience

CERN Summer Student 2025 (full scholarship) (June – August 2025)
Research Topic: DFX4ML: enable Dynamic Function eXchange 4 Machine Learning
Teaching assistant (TA) (2024-2025)
Hardware Synthesis Lab: redefine RISC-V RV32I CPU for student practice, testing the student's lab material, and debugging the student's hardware and coding issues

Skills (Evidence for each skill is provided in the section below)

Hardware description language: Verilog, VHDL
Hardware design: Out-of-Order SuperScalar CPU design
Framework: Chisel3 / Kathryn (my framework) / Gem5 / Pintool/ OpenMp/ Crow/ HLS4ML
High-level Synthesis: Vitis HLS
Programming language (Familiar): C/C++, Python, Java, TCL
Programming language (used to):
Design Software: Autodesk Inventor

Project / Experience

1. **DFX4ML: enable Dynamic Function eXchange for Machine Learning**
   a. using the partial reconfiguration technique to sequentially deploy a partial ML model. Using Magic Architecture to enable an FPGA to reprogram itself without the ease of a CPU.
   b. **Skill:** C/C++, Python, Vitis HLS, Vivado, Verilog, HLS4ML, Partial Reconfiguration

2. **Kathryn: The Hybrid approach for Hardware design, simulation, generation, and profiling**
    a. Abstract hardware design workload, integrated a high-performance hardware simulator and a self profiler, and transpile to Verilog
    b. **Skill:** C/C++, Verilog, Manual Multithreading, Graph representation, Unix Dynamic link, Graph optimization, Hardware abstraction, Vivado TCL
    c. **Link:** https://www.kathryn-tools.org/
3. **Minimum vertex cover solving optimization**
    a. Solve Np(non-polynomial) problem, exploit Multi-threading on graph to find the minimum vertex to cover all consecutive nodes, Winning position compared to my advisor.
    b. **Skill:** C++, OpenMP, Docker, Graph optimization, Parallel task optimization, Bit manipulation, Dynamic programming
    c. **Link:** https://github.com/Tanawin1701d/minimum-vertex-cover
4. **Kathryn_sync**
    a. The cache server is in memory (RAM) and syncs with traditional data, modifies the Crow framework to support large data transfer, optimizes for high-performance transferring, and deploys on Amazon EC2
    b. **Skill:** C/C++/ Python, Intel Vtune (profiler), Crow (networking framework for C++), Graph representation, Manual multithreading, and Data prefetching and run-ahead execution
    c. **Link:** https://github.com/Tanawin1701d/Kathryn_sync, https://github.com/Tanawin1701d/Crow
5. **Simple RISC-V RV32-I assembler**
    a. Transpile the assembly in Scala to a semi-executable file,
    b. **Skill:** Scala, and Bit manipulation
    c. **Link:** https://github.com/Tanawin1701d/simple-RISCV32I-assembler
6. **Simple 4-stage RISC-V RV32-I**
    a. stage pipeline (fetch, decode, execute, and writeback), testing Running merge sort algorithm using the previous assembler
    b. **Skill:** Verilog, C, Computer Architecture
    c. **Link:** https://github.com/Tanawin1701d/riscv-rv32I-multicycle-pipeline

Publication

HoShi: A C++ Framework for High-Performance Hardware Simulation on Single-Core CPU

2025 22nd International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

Award

First position winner in FPGA design training and competition

Link: https://www.facebook.com/share/p/qNgi89mwED9nPjLZ/

References

[1] “The Synchrocyclotron,” *CERN*, Sep. 08, 2025. https://www.home.cern/science/accelerators/synchrocyclotron

CERN SUMMER STUDENT PROGRAMME 2025

รายงานการเข้าร่วมโครงการนักศึกษาภาคฤดูร้อนเซิร์น