Contents

1 Introduction to the Training Establishment **3**

1.1 Company Overview 3

1.2 Company History - Wave Computing 4

1.3 Company History - Paraqum Technologies 5

1.4 Wave-Paraqum partnership, separation and its effect on interns 6

1.5 Organization Structure and Hierarchy 6

1.6 Areas of Interest 7

1.7 Current Situation 8

1.8 Impacts on Sri Lankan Industry 8

1.9 SWOT Analysis 9

1.10 Suggestions to Improve the Company 10

2 Training Experience **11**

2.1 How I got the Opportunity 11

2.2 The Teal Architecture and Wave Flow Graph(WFG) 12

2.2.1 DPU Architecture 12

2.2.2 SoC Perspective 13

2.2.3 Wave design flow 15

2.3 Dive Framework and Wave toolchain 16

2.3.1 Framework Overview 16

2.3.2 Repository Structure 16

2.3.3 WCC 17

2.3.4 WFGsim 18

2.3.5 WFGC 18

2.3.6 WMsim 19

2.3.7 WAsm 19

2.3.8 Current Situation 19

2.4 Py2WFG: A better way to write Wave Flow Graph 20

2.4.1 Python Vs. WFG 20

2.4.2 Python to WFG translator(Py2WFG) . . . 21
2.4.3 Library Structure . . . 22
2.4.4 Python to WFG Simulator . . . 24
2.4.5 DMA engine . . . 25
2.4.6 Unit Testing . . . 26
2.4.7 International Testing . . . 26
2.5 Wave external software tools . . . 28
2.5.1 Overview . . . 28
2.5.2 GitLab . . . 28
2.5.3 Jenkins . . . 29
2.5.4 Confluence . . . 30
2.5.5 Jira . . . 30
2.5.6 Slack . . . 30
2.5.7 Zoom . . . 31
2.5.8 Usage of tools in the SDK verification flow . . . 31
2.6 Life at Wave Computing . . . 32
2.6.1 Daily routine . . . 32
2.6.2 Office facilities . . . 33
2.6.3 Supervisor Visits . . . 33

3 Conclusion **34**
3.1 Training Summary . . . 34
3.2 Self analysis . . . 34
3.3 Verdict on training establishment . . . 35
3.4 Verdict on Industrial training program . . . 36

List of Figures

1.1 Wave Computing logo . . . 3
1.2 Paraqum Technologies logo . . . 3
1.3 Wave Datacenter Server Unit . . . 4
1.4 Wave Consumer Unit . . . 4
1.5 Paraqum Technologies Staff (including Wave team members) [1] . . . 5
1.6 Wave Computing Office location . . . 6
1.7 Wave/Paraqum administration structure . . . 7
1.8 Wave computing Homepage with their target of better AI . . . 7
2.1 Wave DPU with Processing Elements(PEs) . . . 12
2.2 Wave Deep Learning Computer . . . 13
2.3 SoC Perspective of Teal Programmable Accelerator . . . 14
2.4 Data flow structure of Wave Flow Graph designs . . . 15
2.5 Wave Toolchain and design flow . . . 16
2.6 Part of a WaveC code to be compiled on WCC . . . 17
2.7 Part of a WFG code to be compiled on WFGC . . . 18
2.8 Typical Python code snippet . . . 20
2.9 Part of the Py2WFG source code . . . 21
2.10 Example Py2WFG script . . . 22
2.11 Output of the Py2WFG code from Figure 2.10 . . . 23
2.12 A typical IOV file . . . 24
2.13 Py2WFGsim user interface . . . 25
2.14 Wave DPU DMA path . . . 26
2.15 Communications with the overseas testing team . . . 27
2.16 Git protocol functionality . . . 29
2.17 Wave SDK build flow . . . 31
2.18 Typical Wave Computing Workstation . . . 33

List of Tables

Preface

Described in this report is the 24 week training experience I had in wave computing/Paraqum Technologies dating from 25.06.2018 to 07.12.2018 for the fulfillment of industrial training requirements of my B.Sc. in Electronics and Telecommunication Engineering. The report consists of 3 main sections as follows

Chapter 1: Introduction to Wave computing and Paraqum Technologies

Wave computing is a USA based AI startup and Paraqum Technologies is a Sri Lankan Electronic design startup. Wave is now ranked among the top 25 AI providers of the world [6] and Paraqum Technologies is highly respected in Sri Lanka as a pioneer of the industry and recieves design contracts from all over the world. Our training was done through a design contract offered to Paraqum by Wave computing.

Chapter 2: Training Experience

Even though I was an intern, I was given full exposure to the company workflow. I recieved a full scale project to lead and a support team from all over the world to work with. Everyone in the company were very supportive and friendly. The environment was comfortable to work and the workload was perfectly balanced, demanding yet not overwhelming.

Chapter 3: Conclusion

I can say without a doubt, this is one of the best training experiences one can get. The University of Moratuwa Industrial training division, the Department of Electronic and Telecommunication Engineering and National Apprentice and Industrial Training Authority (NAITA) together did their best to provide us with an exceptional experience that will be very useful for the future of our careers.

Acknowledgment

I take this chance to show my gratitude towards everyone that helped make this training possible and facilitated my time with it. First of all I should thank the industrial training division of University of Moratuwa and the National Apprentice and Industrial Training Authority (NAITA) for allowing us to get an industrial training of this magnitude and quality during the time of our studies and always looking out for the wellbeing of the trainees.

I also would like to thank the administration and staff of Paraqum Technologies, starting with the CEO Dr. Ajith Pasqual, Manager Eng. Hasanka Sandujith, Eng. Kasun Tharaka who coordinated the internships and all other staff members who helped in a great many ways to make the training a staggering success.

Last but definitely not least, The staff of the Wave computing team (later separated as Wave computing Sri Lanka) were always dedicated to make the training worthwhile for us. I sincerely thank the Senior Director of Software Engineering division of wave computing Eng. Henrik Esbenson, who visited and supervised us from time to time, General manager of Wave computing Eng. Nuwan Gajaweera, Technical leaders Eng. Upul Ekanayaka and Eng. Binu Amarathunga, My supervisors Eng. Achintha Ihalage, Eng. Omega Gamage and SDK team lead Eng. Dakila Serasinghe and all other staff members of wave computing for everything they did for us.

Chinthana Wimalasuriya,
Undergraduate,
Department of Electronics and Telecommunication Engineering,
University of Moratuwa.

1 Introduction to the Training Establishment

1.1 Company Overview

Wave Computing is a USA Silicon Valley based company that specializes in dataflow technology [3], an alternative to the tensorflow [2] technology used by tech giants such as Nvidia and Google to accelerate AI and neural net training. Although the company is relatively new to the game, they have had a number of notable achievements, which are explained below in detail. They outsource their design contracts to teams around the globe and one of these contracts was undertaken by Paraqum Technologies. This particular team works with the Software development kit (SDK) and the applications of the Wave dataflow platform.

Fig. 1.1. Wave Computing logo

Paraqum Technologies is one of the very first truly Sri Lankan Electronics industry companies in the country. It undertakes Electronic design contracts from big companies from around the world such as Osprey video and of course, Wave Computing. They also have their own network product line. The company had one of their teams working on a design project of Wave Computing which split up in November 2018 to form the Sri Lankan branch of Wave Computing (pvt) Ltd.

Fig. 1.2. Paraqum Technologies logo

My work was focused on the SDK layer of the Wave dataflow platform. This describes a large toolchain of software utilities that are the primary way that an advanced user would interact with the system. This team is also one of the busiest teams of the whole organization because of the sheer complexity of the SDK. In the context of the Sri Lankan staff, the SDK team is the heart of the whole project.

All details disclosed under this section is intellectual property of Wave Computing. I have included details of the training experience as much as possible while abiding with the requirements of Wave Computing on information disclosure.

1.2 Company History - Wave Computing

Wave computing is primarily a project-turned-to-startup by Dr. Derek Meyer, who is also the current CEO. The idea is to design a processor to process the huge amount of data required for complex operations such as large matrix multiplication applications in parallel. He realized his idea in to a semiconductor manufacturing company, wave semiconductor, which has become a legacy as some urls of the company still reads 'wavesemi'. As the prospect of Artificial intelligence became popular, the company understood that their time and effort is best invested in that field and hence, wave semiconductor reformed into Wave Computing AI.

To meet the demand for Artificial intelligence applications, wave computing has planned a series of devices which can fit the requirement of the users whether it is a large datacenter or a small scale business or mobile/IoT devices.

Fig. 1.3. Wave Datacenter Server Unit

Fig. 1.4. Wave Consumer Unit

MIPS Acquisition

MIPS is one of the old time giants of the silicon design game. To facilitate their ideology of integrating dataflow technology to small devices, Wave Computing recently acquired MIPS [5] for their silicon design expertise. With this step, Wave computing hopes to realize their idea of "Bringing datacenter to the edge" by installing datacenter level high performance hardware in small(edge) devices.

1.3 Company History - Paraqum Technologies

Started in 2013 as a continued final year project of four students from the 2008 batch of University of Moratuwa Department of Electronic and Telecommunication Engineering, Paraqum Technologies was first located inside the university itself. The CEO also being a lecturer in the University, this was the result of the attempt to promote technical entrepreneurs to rise up from the university level.

Paraqum Technologies quickly developed into a full sized technology startup and by 2018, the staff had grown to around 40 and every year they took around 10-12 interns under their wings to properly expose them to the electronic industry. Now they have design contracts from renowned industrial leaders and their own networking product line which is also very famous for their unique capabilities.

Fig. 1.5. Paraqum Technologies Staff (including Wave team members) [1]

1.4 Wave-Paraqum partnership, separation and its effect on interns

As described above, Paraqum Technologies had a design contract from wave computing for their toolchain and application needs. This contract had been in place for almost two years and has proved to be one of the most productive teams wave computing has ever employed. They have helped in designing the hardware part of the dataflow processing chips and by the time we started our internships, they were working on RTL level software projects.

However, Wave computing had decided it would be better to acquire the Sri Lankan team for themselves. So, in november 2018, the Wave Computing team was separated from Paraqum Technologies and established as the Sri Lankan branch of Wave Computing (pvt) ltd. with no connection to Paraqum Technologies whatsoever. Before Separation, Wave computing team worked in the Paraqum Technologies office in Kohuwala but after that they moved to a new office in Bambalapitiya.

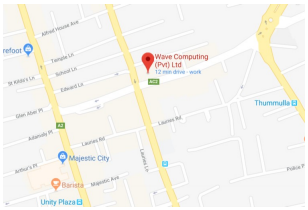


Fig. 1.6. Wave Computing Office location

As the organizations separated, interns of the wave team also moved to work in the new office premises and work was carried out as normal. But the internship contracts were not changed and technically, we were interns of Paraqum Technologies for the whole 24 weeks but worked under the supervision of Wave Computing staff. Therefore this report will be more focused on Wave Computing.

1.5 Organization Structure and Hierarchy

Wave computing team was another division in Paraqum Technologies until the separation but after that they became a fully independent entity. The dotted line in the following diagram represents the administration link that existed before the separation. Now Wave computing Sri Lanka

operates directly under the administration of their head office in Campbell, California.

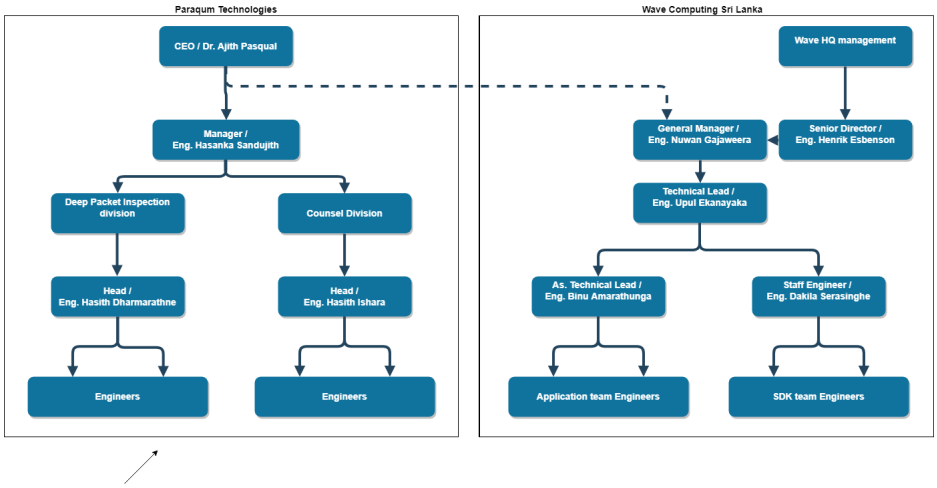


Fig. 1.7. Wave/Paraqum administration structure

1.6 Areas of Interest

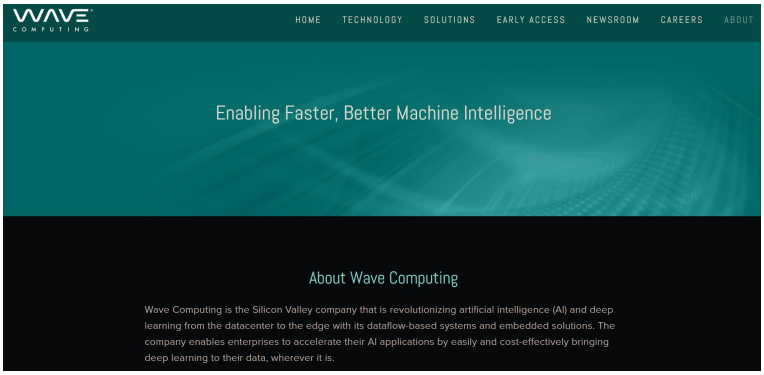


Fig. 1.8. Wave computing Homepage with their target of better AI

The main focus of Wave computing is to Introduce new processing system for heavy computational operations with their proprietary dataflow processing technology. When the system is built and released, it can be used for:

Artificial Intelligence

The main use of the dataflow processing units (DPU) is intended to be Artificial Intelligence systems. Current hardware in our devices are not suited for the massive amount of data that need

to be processed in parallel. The DPU has been optimized for this particular task to perform it at high speed. It is speculated that once wave DPU systems become available in market, it will spark a revolution in the AI industry.

Edge devices

Edge devices are small devices such as mobile phones and IoT sensors. Wave computing already has plans in motion to develop high performance, small size hardware to be installed in these edge devices. This will allow us to have very powerful yet small devices for our day to day usage.

Image processing

Image processing and AI are similar in many ways. The parallel data handling capacity of DPU systems makes it an ideal candidate for image processing applications such as self driving or biometric authentication.

1.7 Current Situation

The initial release of Wave hardware to the general public is scheduled for 2020 and the company is doing well on the roadmap to meet this target. The hardware design is nearly perfect and the software is halfway through. It can be expected that the company will meet the target soon enough and they are expanding rapidly now, with the new office in Sri Lanka hiring more and more people.

1.8 Impacts on Sri Lankan Industry

Having recently completed a very successful series E funding round [7], Wave Computing has a fairly large capital at its disposal and are ready to invest well in the Sri Lanka division. This huge amount of money will directly enter Sri Lankan economy in US dollars, aiding the stabilization of economy. They also offer employment to talented local engineers in bigger and bigger numbers every year. Finally, when the Wave product line enters market and becomes a key element in the AI game, Sri Lankans will have an important role to play in it, carrying us forward in the technical development sector.

1.9 SWOT Analysis

Strengths

- Wave already has MIPS under them, giving them a considerable head start on the silicon game. They got a team of experienced engineers with this acquisition.
- They have teams around the world with different perspectives, allowing creative solutions to problems.
- The company is very well funded, giving them plenty of runway.
- The startup culture is well maintained in the workplaces, making it appealing to employees.

Weaknesses

- Without any kind of product on the market yet, the company solely runs on VC funding
- Global teams sometimes cause delays in communications.

Opportunities

- AI field is expanding day by day, meaning the potential market for the company widens continuously.
- Another funding round can result in more and more funds due to the promising nature of the project.
- Reputation of the company is luring in more talent as it expands.
- Since the hardware is not restricted only to AI applications, there can be more potential uses to the devices.

Threats

- There may be rival companies with better solutions to the AI processing problem than the DPU technology.

- Some of the global teams are from politically and economically unstable countries and they might be lost to national crisis.
- DPU is not fully built yet. It may not give the expected outcomes and performance in real world.

1.10 Suggestions to Improve the Company

- Some sort of outbound activities can be organized to allow the team to bond even better.
- Interns could be provided with some more training sessions.

2 Training Experience

2.1 How I got the Opportunity

A long time before the industrial training selection, I had heard Paraqum Technologies was one of the best places to learn about the electronics industry in Sri Lanka and Its CEO, Dr. Ajith Pasqual had taught a module in the university that sparked an interest in me about the subject of silicon design. Therefore, when Paraqum Technologies was listed as an open CV company, I did not hesitate to submit a CV, which got selected by the company staff who then interviewed me thoroughly in their office and sometime later, informed me that I had been selected to the Wave Computing Division.

At the start of the Internship, I was placed under the supervision of Eng. Achintha Ihalage, an application engineer whose original task was to handle the timing and constraints of the DPU chip. He walked me through the basics of setting up the Wave Computing workspace, company work ethics and other needed technical skills. He also introduced us to the DPU hardware and other proprietary technologies by Wave.

During the second week, Eng. Henrik Esbenson, who is in charge of the Sri Lankan team at Wave HQ visited the office and demonstrated his ideas for projects. One of these was the Python - Wave Flow Graph translator, also known as Py2WFG. I volunteered to take that project and Eng. Henrik allocated me the necessary resources of the company including support teams and software tools to carry on the project.

This project soon became popular among the crowd of wave computing and I developed it according to the requirements and feedback. The amount of work was rather large but I managed to complete the project and hand it over by the time Internship ended. This project will most likely be adopted by full time developers and expanded to probably replace the existing design flow.

2.2 The Teal Architecture and Wave Flow Graph(WFG)

Teal is essentially the best of both programmable logic platforms and Application Specific Integrated Circuits(ASIC) combines together to give out the best performance and power efficiencies. In fact both hardware and software designs can be transferred onto the Teal fabric. Basically, Teal can be broken down into smallest unit of a Processing Element(PE). This is a simple processing unit which can carry out 8 bit operations adhering to Reduced Instruction SET (RISC) architecture.

2.2.1 DPU Architecture

A Processing Element will inherit an 8 bit accumulator who in return is connected to its neighboring Processing Elements. It is evident that a combination of a large number of such Processing Elements achieve as much parallelism as witnessed in the computational design history of the world. This revolutionary design with extensively parallel computational ability is a completely ground-breaking approach to the existing technologies used in chips for parallelism. Incidentally, a combination of 16 Processing Elements make up for a Cluster and a combination of 64 Clusters form a Super-Cluster and 16 such Super-Clusters form a single Wave Dataflow Processing Unit. The projected design for this massively complex structure has now reached the taping out process an it will only be a matter of time before the first chip gets manufactured physically. Such a developed Processing Element is projected to have a speed of close to 10GHz.

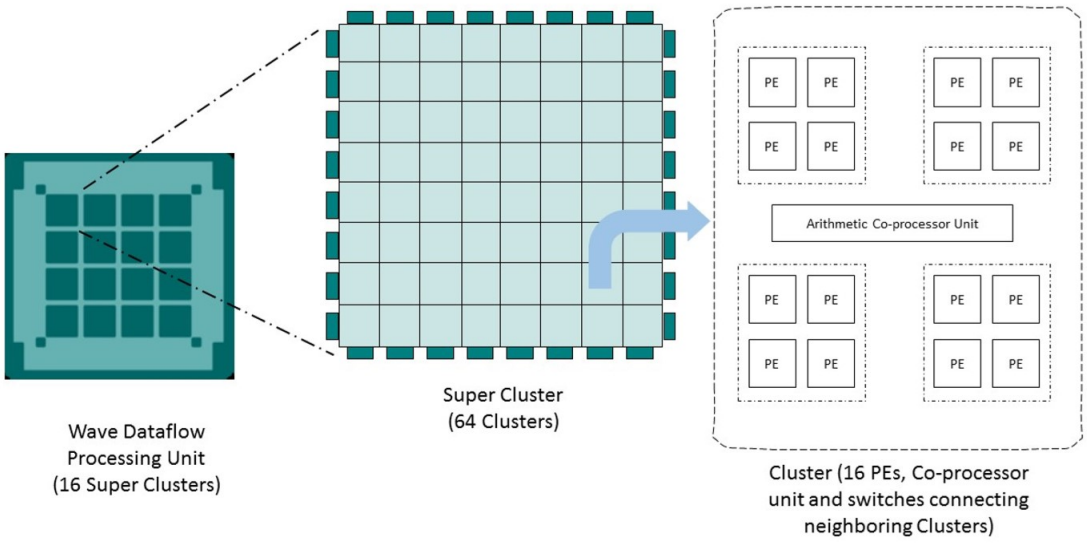


Fig. 2.1. Wave DPU with Processing Elements(PEs)

The Figure 2.1 is a comprehensive illustration of the structure of Teal design. It is noteworthy to realize that each Processing Element acts much similar to a single processing unit and therefore, each Super Cluster can be viewed in the perspective of thousands of processing units ready to function simultaneously. Nevertheless, the true power of Wave Dataflow Processing Unit design is shown in Figure 2.2 where the extent of operation of the final design is depicted. The final design is projected to have a whopping 2 Peta Operations per second amount of power which is ideally suited for the needs of computational power of the future. It is safe to say that Wave Dataflow Processing Unit will become by far the fastest ever processing unit of the world.

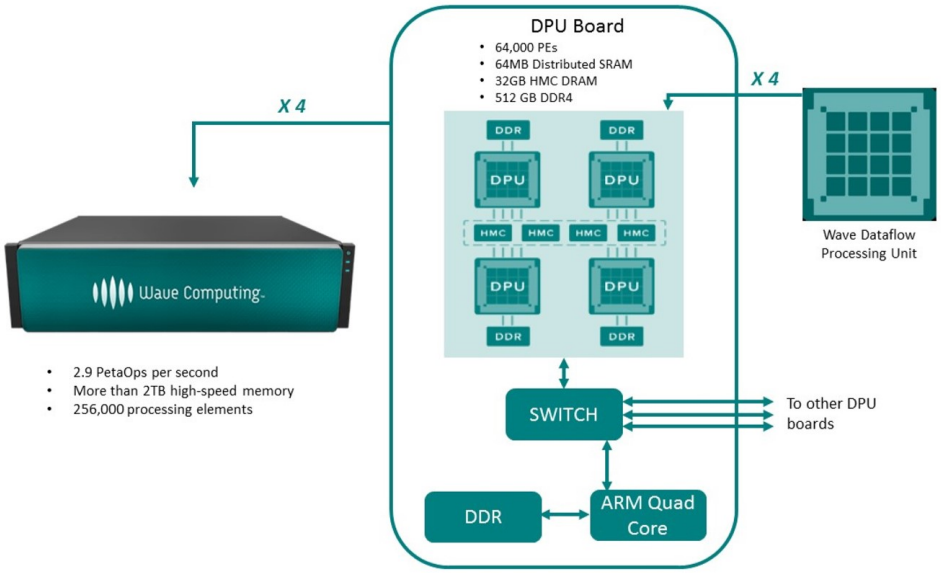


Fig. 2.2. Wave Deep Learning Computer

2.2.2 SoC Perspective

The Teal processing accelerator is essentially integrated into a System-on-a-Chip(SoC). As shown in the Figure 2.3 it is connected to processor SoC using streaming AXI interfaces. It has a direct connection toward high speed IO streams. The DMA controller (refer Figure 2.3) is the key connector between external DRAM internal Block RAMs of Teal. The accelerator itself is demand driven, meaning it will stay at sleep state unless instructed otherwise. The importance of this viewpoint was important to me as some of the design work carried out were finding a way to simulate DMA inputs and outputs. Since accessing data from external RAMs is through DMA 'Channels', some of the work related to improvements in the DMA engine were carried out during

my project executions.

The next interesting design feature of Wave Data-flow processing model are the Wave Flow Graphs. This is a representation scheme for which operations are represented as nodes and values as edges. Nodes and edges connect together in a network of operations to perform a computational task. The idea behind the clock-less operation of the tasks come through the ability for nodes to perform the intended operation whenever valid inputs are provided. A dedicated clock is not needed and thus presence of valid data at the inputs automatically will define trusted operation.

However, a sense of timing is available in the form of the concepts of tics and sub-tics. A tic is equivalent to a circular round of operations executed by a Processing Element buffer which is currently defined as 256 instructions. In fact, it is 256 times the instruction cycle time. Due to the fact that instruction cycle time is not specifically designated but rather dependent on the execution of each operations' processing speed, it is not fair by the Wave DPU design to have a specific tic-rate. Incidentally, there is a typical rate of 10 GHz for a sub-tic cycle. a sub-tic cycle is essentially the rate at which an instruction from a circular buffer(IRAM) is fetched, processed and switched.

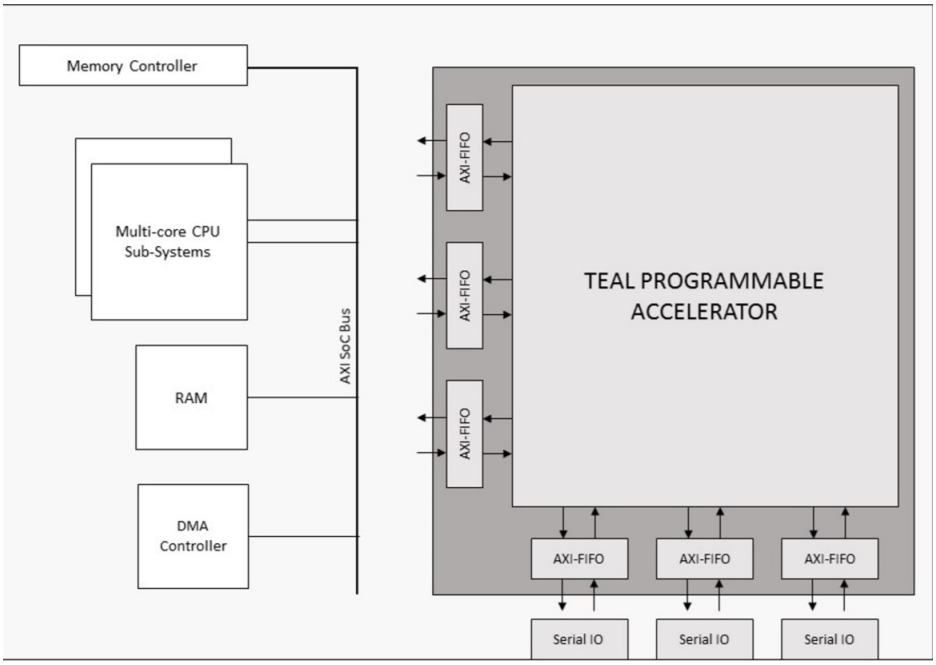


Fig. 2.3. SoC Perspective of Teal Programmable Accelerator

Apart from the contrast between the parallelized implementation of Wave Flow Graph designs and the continuous flow of operations within PEs, Wave Flow Graph is a defined rule to represent any program within the Teal fabric. In section 2.4.6 (Unit Testing) a direct implementation of testing the Wave Flow Graph operations carried out by me, is explained in detail.

2.2.3 Wave design flow

Owing to the complexity of parallel PE operation it is evidently clear that programming instructions into the Teal Architecture is not achievable manually. Therefore, an EDA flow is prevalent so that any hardware design can be brought down through compilation, scheduling, mapping and routing to the Teal fabric level. This flow of compilation from hardware design to Byte-fabric level is done through the Wave introduced design flow. The Figure 2.4 is a representation of a very simple Wave Flow Graph design where you can see the nature of operation of the language. It basically consists of bit wise operations between inputs and outputs. Most operations are related to bit level manipulation of data and it is note worthy to realize that these operations are carried out for 8 bit data chunks. The combination of several bytes of data can be processed with special modules of Wave Flow Graph code where the data is subdivided and processed accordingly. The node edge representation of the design will give you enough ides as to the complexity that can arise with the introduction of several operations and variables into one design. The final structure will be a collection of various nets combined together from input end to output end.

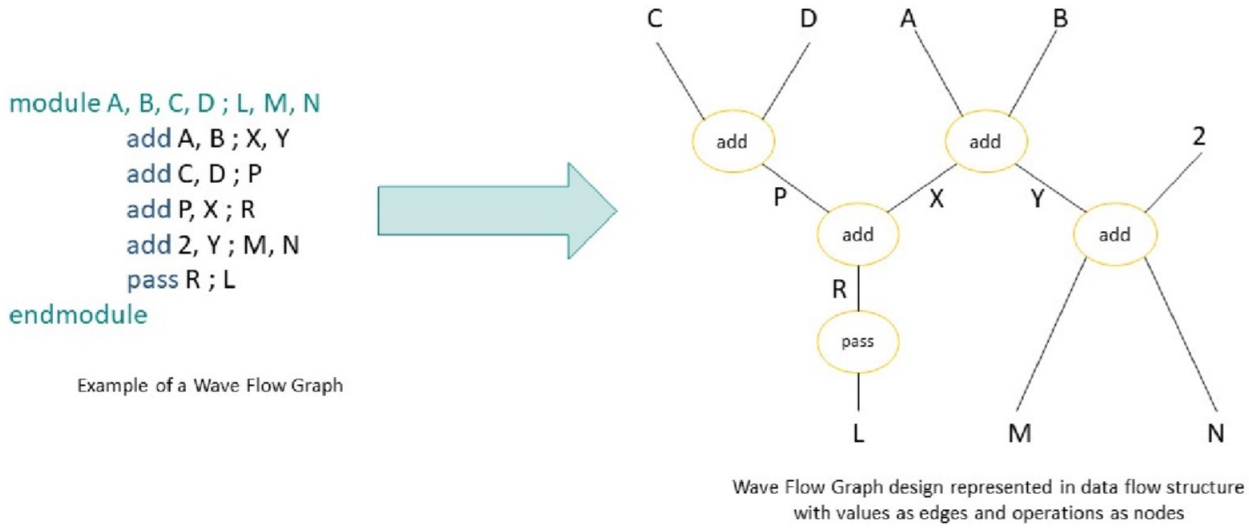


Fig. 2.4. Data flow structure of Wave Flow Graph designs

2.3 Dive Framework and Wave toolchain

2.3.1 Framework Overview

When writing a new design for the DPU, the engineers use an in-house written API called the dive framework for testing and Verification. Due to the sheer complexity of the programs that need to be adapted to run in the DPU, the code is brought down to lower semantic levels step by step using the toolchain which consists of 5 main tools.

- Wave C Compiler (WCC) - Compiles the C++ code to a WFG code
- Wave Flow Graph Simulator (WFGsim) - Simulates the flow graph and compares it with the I/O values of the C++ code
- Wave Flow Graph Compiler (WFGC) - Compiles the WFG code to a Wave Assembly code
- Wave Machine Simulator (WMsim) - Simulates the Wave assembly code in a virtual machine environment
- Wave Assembly Compiler (WAsm) - Compiles the assembly code to an encrypted binary file format called Lantana

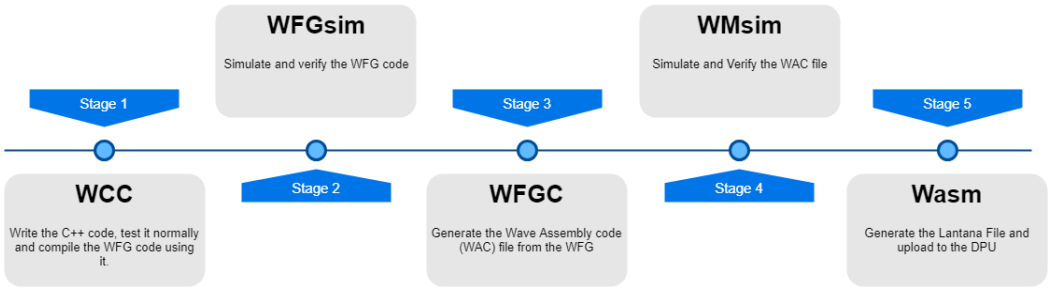


Fig. 2.5. Wave Toolchain and design flow

2.3.2 Repository Structure

Different cloud folders that contains software code are known as repositories. Wave source codes are stored in five main repositories hosted in a local git server. For better identification, the tools are again divided in to two classes, Wave Front End (WFE) and Wave Back End (WBE).

WCC, WFGsim and WFGC belong to the WFE while WMsim and WAsm are contained in the WBE. In the Wave source code, the code for WFE and WBE are stored in separate repositories while the code that is shared by these are stored in a third repository named wcore. Test codes are stored in the wtest repository while experimental tools are stored in the final repository, tools.

2.3.3 WCC

```
typedef short int16_t;
typedef unsigned short uint16_t;
typedef char int8_t;
typedef unsigned char uint8_t;

#define A 10
#define B 100
__attribute__ ((noinline))
void module(uint8_t start,
            uint8_t parameter01,
            int16_t parameter02[20],
            uint8_t *stop,
            volatile int32_t *memory01,
            volatile int32_t *memory02){

static uint8_t start_reg;
static uint8_t parameter01_reg;
static uint8_t value_reg;
uint8_t value, param;
if (cond){
    parameter01_reg = value;
}
if (value & param){
    *stop=1;
}
else {
    *stop = 0;
}
int32_t array[10];
for(int8_t i=0;i<10;i=i+1){
    array[i]=_conc4(#,#,#,#,_extra2(#,parameter02[2*i+1]),_extra2(#,parameter02[2*i+1]),_extra2(#,parameter02[2*i]),_extra2(#,parameter02[2*i]));
}
if (value) memory01[_conc2(0x00,param)] = B ;
if (cond){
    start_reg = 1;
}
else if (*stop){
    start_reg=0;
}
uint8_t bin[8] ;
bin[0] = 0x00 ;
bin[1] = 0x00 ;
```

Fig. 2.6. Part of a WaveC code to be compiled on WCC

Wave computing has their own version of C++ dubbed WaveC, which is C++ with added support for data channels and some more things. This language receives constant updates from a team inside wave and thus, developers must build the C language from its source code frequently. This is also true for other tools.

WCC uses a technology called LLVM [4] to convert one human written code in to a different human readable code. It works by analyzing the code and simplifying it to a desired level and assembling the result back together on a predefined syntax. After simplifying the code again and again, the desired output can be obtained, which in this case is the WFG code file.

2.3.4 WFGsim

This was the most important tool for me since my project could eventually evolve to fully replace this tool. I had to analyze this tool very thoroughly to extract functionality for the Py2WFG simulator (section 2.4.4).

WFGsim works by multi-thread programming on python. It initializes separate threads for DPU sections and host processors. Inside each python thread, a library called boost is used to invoke a linear C++ script. These scripts 'talk' to each other via the python interface and python interface also gives the DPU sections a sense of time by governing the order which operations are executed. This process is continued until the necessary number of 'tics' (see section 2.2.2) has elapsed and after that, the results are presented to the user in a file.

2.3.5 WFGC

```
'ldi 0, #; _pre2 : &mem=reg1
'xorop _b, not_tobool; tmpo
'sto 0, tmp, 1; dep : &mem=reg2
'mui tobool, _pre, #; tmp4
'cne go, Qa; inv_tobool
'sto 0, offst_addr, inv_; *, * : &mem=reg6
'mux tobool, _pre2, offset_addr_0; tmp5
'sto 0, offst_addr, inv_; * : &mem=reg5
'ldi 0; tmp7, * : &mem=reg_add
'ldi 0; tmp8, * : &mem=reg_add
add_x_ tmp5, tmp6, tmp7, tmp8; A, B
add_x_ tmp5, tmp6, 0, 7; add81, add_0
add_x_ add_1, add_0, tmp4, tmp5; tmp10_1, tmp10_0
add_x_ tmp7, tmp8, 0, 2; add15_1, add15_0
ldin_4_w_pe_3 tmp7, , __din_mem_neu1; tmp11_3, tmp11_2, tmp11_1, tmp11_0, * : $mem=host1
ldin_4_w_pe_3 tmp9_1, , __din_mem_syn1; tmp12_3, tmp12_2, tmp12_1, tmp12_0 : $mem=host2
ldin_4_w_pe_3 tmp7, , __din_mem_neu1e; tmp13_3, tmp13_2, tmp13_1, tmp13_0 : $mem=host1
ldin_4_w_pe_3 add_1, , __din_mem_neu1; tmp14_3, tmp14_2, tmp14_1, tmp14_0, * : $mem=host2
ldin_4_w_pe_3 tmp10_1, , __din_mem_syn1; tmp15_3, tmp15_2, tmp15_1, tmp15_0, tmp15_dep : $mem=host3
ldin_4_w_pe_3 add_0, __din_mem_neu1e; tmp16_3, tmp16_2, tmp16_1, tmp16_0, tmp16_dep : $mem=host3
sim tmp12_1, tmp12_0, tg_1, tg_0; tmp20_3, tmp20_2, tmp20_1, tmp20_0
scale_32_32_b_wo tmp20_3, tmp20_2, tmp20_1, tmp20_0, 14; spneu_0_3, spneu_0_2, spneu_0_1, spneu_0_0
cast_high tmp13_1, tmp13_0; neu1e_1_0_int_3, neu1e_1_0_int_2, neu1e_1_0_int_1, neu1e_1_0_int_0
'pass spneu_0_3; tmp24_3
'pass spneu_0_2; tmp24_2
'pass spneu_0_1; tmp24_1
'pass spneu_0_0; tmp24_0
'pass neu1e_1_0_int_3; tmp25_3
'pass neu1e_1_0_int_2; tmp25_2
'pass neu1e_1_0_int_1; tmp25_1
'pass neu1e_1_0_int_0; tmp25_0
add_x_1 tmp24_3, tmp24_2, tmp24_1, tmp24_0, tmp25_3, tmp25_2, tmp25_1, tmp25_0; tmp26_3, tmp26_2, tmp26_1, tmp26_0
con_32_16_b_rx tmp26_3, tmp26_2, tmp26_1, tmp26_0; spt_neu_0_st_1, spt_neu_0_st_0
simul tmp12_3, tmp12_2, tg_1, tg_0; tmp30_3, tmp30_2, tmp30_1, tmp30_0
scale_32_32_b_wo tmp30_3, tmp30_2, tmp30_1, tmp30_0, 14; spneu_1_3, spneu_1_2, spneu_1_1, spneu_1_0
cast_high tmp13_3, tmp13_2; neu1e_1_1_int_3, neu1e_1_1_int_2, neu1e_1_1_int_1, neu1e_1_1_int_0
'pass spneu_1_3; tmp34_3
'pass spneu_1_2; tmp34_2
'pass spneu_1_1; tmp34_1
'pass spneu_1_0; tmp34_0
'pass neu1e_1_1_int_3; tmp35_3
'pass neu1e_1_1_int_2; tmp35_2
```

Fig. 2.7. Part of a WFG code to be compiled on WFGC

WFGC also uses a LLVM to bring the WFG code (Figure 2.7) down another step on the semantic level. The resulting WAC file is a little bit similar to standard assembly and is human readable. But due to the facts that it is designed to operate on a complex array of processors and has numerous

timing constraints makes it hard to understand how the code exacly works. So this WAC code is used only to make very deep level optimizations.

2.3.6 WMsim

WMsim simulates the WAC file in a manner similar to WFGsim. This step is used to verify the design is bug free before loading it onto the actual hardware.

2.3.7 WAsm

WAsm is not yet used very much since the hardware is not finalized. But what it does is it generates a binary file which is encrypted by AES 256 military encryption to make it tamper proof. This code has the instructions for wiring the PEs together, Data steering, power management, communication and other constraints bundled with the actual program instructions. Once this is uploaded, the DPU becomes ready to use.

2.3.8 Current Situation

Although the tools are being used, they are still slightly buggy. So, most of the time, LLVM outputs are not perfect. Therefore, engineers spend a lot of time editing the outputs to make their respective simulators pass the designs. This is currently the task of most of the engineers in the application team.

2.4 Py2WFG: A better way to write Wave Flow Graph

2.4.1 Python Vs. WFG

```python
class job():
    def __init__(self, payment, start, end):
        self.payment_ = int(payment)
        self.start_ = int(start)
        self.end_ = int(end)

def get_end_day(jobs):
    res = 1

    for j in jobs:
        if j.end_ > res:
            res = j.end_

    return res
```

Fig. 2.8. Typical Python code snippet

The Figure 2.8 shows a typical python code example and Figure 2.4 shows a WFG snippet. These two languages were built for two entirely different purposes but the highly adaptable nature of python makes a valid point whether if it can replace the functionality of WFG. But both languages have their pros and cons.

Python

Python is a general purpose scripting language which focuses on user friendliness. It supports object oriented programming and is also backed up by a huge number of highly optimized libraries for various purposes. But when compared with languages such as C++ and Java, Python is heavy on the memory and slow for large volume computing. It is also not optimized for the particular purpose of programming the Wave DPU.

WFG

WFG was created with one purpose and one purpose only in mind, Programming the Wave DPU. Thus it has primary operators that can precisely match the deep capabilities of the DPU hardware. It can be very efficient too when properly programmed. The downside to this language is that it is

very strongly typed and is a user friendliness nightmare. Repetitive operators need to be manually entered by the user and the language has no capability of any kind of looping of the language itself.

2.4.2 Python to WFG translator(Py2WFG)

A solution was created by putting the best of both worlds together. The idea is to create a python 'sleeve' to cover up the WFG interface and present the user with a way to interact with python and get WFG level results. The user will now write up the script and when he runs it, it will output a fully optimized WFG script.

```
55  class wNet():···
104
105 class wAttrib():···
116
117
118 class BasePrimMod(): # abstract base class
119     def __init__(self, name, inputs, outputs):
120         self.name_ = name.strip() if type(name) == str else name
121
122         self.inputs_ = []
123         self.outputs_ = []
124         self.InputsSaturated_ = False
125         self.OutputsSaturated_ = False
126         self.debug = False
127         self.Memory_ = None
128
129         if type(inputs) != list:···
131
132         if type(outputs) != list:···
134
135         for i in inputs:···
142
143         for o in outputs:···
149
150         self.UserAttr_ = []
151
152     def __del__(self):···
155
156     def attr(self, attrDict):···
179
180     def GetAttrStr(self):···
```

Fig. 2.9. Part of the Py2WFG source code

Code Translation

The basic idea for this library is derived from Tensorflow [2], which is a python library that allows easy neural net design using a python library. The user will still need a sufficient knowledge on WFG syntax to use the library. But it will eliminate the hassles of writing a slightly-above-assembly language scripts by hand. The script that user needs to write is similar to the one showed in Figure 2.10. The library needs to be imported in to the script, Then Modules can be created

from the wMod class in the library. These module 'husks' are then filled with dataflow operations as needed. When complete, issuing the makeWFG command will write a full WFG script that can be later integrated into the existing wave design flow.

```
import wave.py2wfg as p

dt_16_k = p.wMod('dt_16_k')

din = dt_16_k.AddBus('din', 16)
dout = dt_16_k.AddBus('dout', 16)

inputs = dt_16_k.Input(['full', 'empty'] + din)
outputs = dt_16_k.Output(['get', 'put'] + dout)

xor1 = dt_16_k.wxor('full', '0x01', 'not_full').name('checkfull')
xor2 = dt_16_k.wxor('empty', '0x01', 'not_empty').name('checkempty')

and1 = dt_16_k.wand('not_empty', 'not_full', 'put').name('checkget')

or1 = dt_16_k.wor('0x00', '0x01', 'get')
or2 = dt_16_k.opBank('or', 16, [din, '0x00'], [dout])

for i in range(16):
    or2[i].name('dtran'+str(i))

dt_16_k.makeWFG('dt_16_k')
```

Fig. 2.10. Example Py2WFG script

Running the example script shown in Figure 2.10 will output a WFG file that contains the operations represented in the script, which is shown in Figure 2.11

This output WFG script unlike the handwritten scripts, is properly formatted and can be reconfigured easily through the python script again. This ease of use further improved with the introduction of the Simulator (section 2.4.4).

2.4.3 Library Structure

Data Types

Main data types used in modeling a WFG module using Py2WFG are,

- Primitive

```
'module dt_16_k full, empty, din_0, din_1, din_2, din_3, din_4, din_5, din_6,
din_7, din_8, din_9, din_10, din_11, din_12, din_13, din_14, din_15; get, put,
dout_0, dout_1, dout_2, dout_3, dout_4, dout_5, dout_6, dout_7, dout_8, dout_9,
dout_10, dout_11, dout_12, dout_13, dout_14, dout_15

      'xor full, 0x01;  not_full : name=checkfull
      'xor empty, 0x01;  not_empty : name=checkempty
      'and not_empty, not_full;  put : name=checkget
      'or 0x00, 0x01;  get
      'or din_0, 0x00; dout_0: name=dtran0
      'or din_1, 0x00; dout_1: name=dtran1
      'or din_2, 0x00; dout_2: name=dtran2
      'or din_3, 0x00; dout_3: name=dtran3
      'or din_4, 0x00; dout_4: name=dtran4
      'or din_5, 0x00; dout_5: name=dtran5
      'or din_6, 0x00; dout_6: name=dtran6
      'or din_7, 0x00; dout_7: name=dtran7
      'or din_8, 0x00; dout_8: name=dtran8
      'or din_9, 0x00; dout_9: name=dtran9
      'or din_10, 0x00; dout_10: name=dtran10
      'or din_11, 0x00; dout_11: name=dtran11
      'or din_12, 0x00; dout_12: name=dtran12
      'or din_13, 0x00; dout_13: name=dtran13
      'or din_14, 0x00; dout_14: name=dtran14
      'or din_15, 0x00; dout_15: name=dtran15

'endmodule
```

Fig. 2.11. Output of the Py2WFG code from Figure 2.10

- Pseudo
- Module
- Net

Primitive

Primitives are basically WFG dataflow operations such as and, or, xfers, channel. These usually have inputs, outputs and attributes, depending on the specific primitive. These are the primary building blocks of a module.

Pseudo

These are used for depicting memory operations such as ram and rom. They have connections different from dataflow operations but still have attributes just like dataflow.

Module

Primitives and Memory get together to form modules. A module can correspond to a single WFG file or in more complex designs, several instances of one module can be created inside another module to form a hierarchical design.

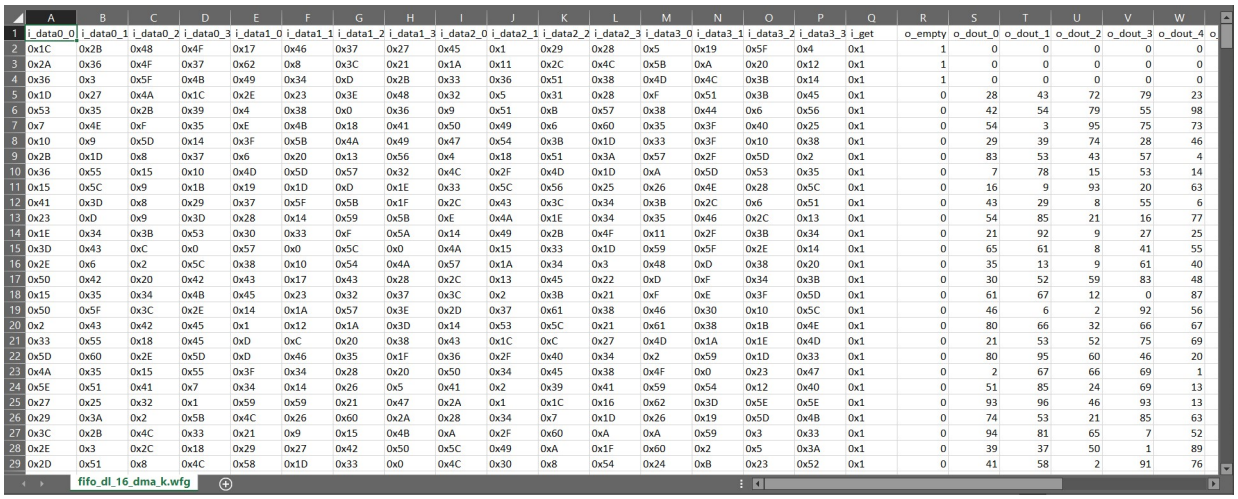
Net

These are somewhat similar to wires. They are used to interconnect modules and primitives. A collection of nets, grouped together for a common purpose is known as a bus.

2.4.4 Python to WFG Simulator

The simulation of flow graph files are now done through WFGsim (section 2.3.4). This platform is very complex and simulating a design on it takes a lot of overhead. To avoid this, an idea came up to add the ability to simulate Flow Graph designs on the Py2WFG library itself. This would allow the user to quickly write up a design on python and without exiting the python shell, they could quickly and efficiently simulate the design.

Before calling up the simulator, if you need to test your designs against custom inputs, you need to define them in a special file called a input/output vector (iov) file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	i_data0_0	i_data0_1	i_data0_2	i_data0_3	i_data1_0	i_data1_1	i_data1_2	i_data1_3	i_data2_0	i_data2_1	i_data2_2	i_data2_3	i_data3_0	i_data3_1	i_data3_2	i_data3_3	i_get	o_empty	o_dout_0	o_dout_1	o_dout_2	o_dout_3	o_dout_4	o
2	0x1C	0x2B	0x48	0x4F	0x17	0x46	0x37	0x27	0x45	0x1	0x29	0x28	0x5	0x19	0x5F	0x4	0x1	1	0	0	0	0	0	
3	0x2A	0x36	0x4F	0x37	0x62	0x8	0x3C	0x21	0x1A	0x11	0x2C	0x4C	0x5B	0xA	0x20	0x12	0x1	1	0	0	0	0	0	
4	0x36	0x3	0x5F	0x4B	0x49	0x34	0xD	0x2B	0x33	0x36	0x51	0x38	0x4D	0x4C	0x3B	0x14	0x1	1	0	0	0	0	0	
5	0x1D	0x27	0x4A	0x1C	0x2E	0x23	0x3E	0x48	0x32	0x5	0x31	0x28	0xF	0x51	0x3B	0x45	0x1	0	28	43	72	79	23	
6	0x53	0x35	0x2B	0x39	0x4	0x38	0x0	0x36	0x9	0x51	0xB	0x57	0x38	0x44	0x6	0x56	0x1	0	42	54	79	55	98	
7	0x7	0x4E	0xF	0x35	0xE	0x4B	0x18	0x41	0x50	0x49	0x6	0x60	0x35	0x3F	0x40	0x25	0x1	0	54	3	95	75	73	
8	0x10	0x9	0x5D	0x14	0x3F	0x5B	0x4A	0x49	0x47	0x54	0x3B	0x1D	0x33	0x3F	0x10	0x38	0x1	0	29	39	74	28	46	
9	0x2B	0x1D	0x8	0x37	0x6	0x20	0x13	0x56	0x4	0x18	0x51	0x3A	0x57	0x2F	0x5D	0x2	0x1	0	83	53	43	57	4	
10	0x36	0x55	0x15	0x10	0x4D	0x5D	0x57	0x32	0x4C	0x2F	0x4D	0x1D	0xA	0x5D	0x53	0x35	0x1	0	7	78	15	53	14	
11	0x15	0x5C	0x9	0x1B	0x19	0x1D	0xD	0x1E	0x33	0x5C	0x56	0x25	0x26	0x4E	0x28	0x5C	0x1	0	16	9	93	20	63	
12	0x41	0x3D	0x8	0x29	0x37	0x5F	0x5B	0x1F	0x2C	0x43	0x3C	0x34	0x3B	0x2C	0x6	0x51	0x1	0	43	29	8	55	6	
13	0x23	0xD	0x9	0x3D	0x28	0x14	0x59	0x5B	0xE	0x4A	0x1E	0x34	0x35	0x46	0x2C	0x13	0x1	0	54	85	21	16	77	
14	0x1E	0x34	0x3B	0x53	0x30	0x33	0xF	0x5A	0x14	0x49	0x2B	0x4F	0x11	0x2F	0x3B	0x34	0x1	0	21	92	9	27	25	
15	0x3D	0x43	0xC	0x0	0x57	0x0	0x5C	0x0	0x4A	0x15	0x33	0x1D	0x59	0x5F	0x2E	0x14	0x1	0	65	61	8	41	55	
16	0x2E	0x6	0x2	0x5C	0x38	0x10	0x54	0x4A	0x57	0x1A	0x34	0x3	0x48	0xD	0x38	0x20	0x1	0	35	13	9	61	40	
17	0x50	0x42	0x20	0x42	0x43	0x17	0x43	0x28	0x2C	0x13	0x45	0x22	0xD	0xF	0x34	0x3B	0x1	0	30	52	59	83	48	
18	0x15	0x35	0x34	0x4B	0x45	0x23	0x32	0x37	0x3C	0x2	0x3B	0x21	0xF	0xE	0x3F	0x5D	0x1	0	61	67	12	0	87	
19	0x50	0x5F	0x3C	0x2E	0x14	0x1A	0x57	0x3E	0x2D	0x37	0x61	0x38	0x46	0x30	0x10	0x5C	0x1	0	46	6	2	92	56	
20	0x2	0x43	0x42	0x45	0x1	0x12	0x1A	0x3D	0x14	0x53	0x5C	0x21	0x61	0x38	0x1B	0x4E	0x1	0	80	66	32	66	67	
21	0x33	0x55	0x18	0x45	0xD	0xC	0x20	0x38	0x43	0x1C	0xC	0x27	0x4D	0x1A	0x1E	0x4D	0x1	0	21	53	52	75	69	
22	0x5D	0x60	0x2E	0x5D	0xD	0x46	0x35	0x1F	0x36	0x2F	0x40	0x34	0x2	0x59	0x1D	0x33	0x1	0	80	95	60	46	20	
23	0x4A	0x35	0x15	0x55	0x3F	0x34	0x28	0x20	0x50	0x34	0x45	0x38	0x4F	0x0	0x23	0x47	0x1	0	2	67	66	69	1	
24	0x5E	0x51	0x41	0x7	0x34	0x14	0x26	0x5	0x41	0x2	0x39	0x41	0x59	0x54	0x12	0x40	0x1	0	51	85	24	69	13	
25	0x27	0x25	0x32	0x1	0x59	0x59	0x21	0x47	0x2A	0x1	0x1C	0x16	0x62	0x3D	0x5E	0x5E	0x1	0	93	96	46	93	13	
26	0x29	0x3A	0x2	0x5B	0x4C	0x26	0x60	0x2A	0x28	0x34	0x7	0x1D	0x26	0x19	0x5D	0x4B	0x1	0	74	53	21	85	63	
27	0x3C	0x2B	0x4C	0x33	0x21	0x9	0x15	0x4B	0xA	0x2F	0x60	0xA	0xA	0x59	0x3	0x33	0x1	0	94	81	65	7	52	
28	0x2E	0x3	0x2C	0x18	0x29	0x27	0x42	0x50	0x5C	0x49	0xA	0x1F	0x60	0x2	0x5	0x3A	0x1	0	39	37	50	1	89	
29	0x2D	0x51	0x8	0x4C	0x58	0x1D	0x33	0x0	0x4C	0x30	0x8	0x54	0x24	0xB	0x23	0x52	0x1	0	41	58	2	91	76	

fifo_dl_16_dma_k.wfg

Fig. 2.12. A typical IOV file

The values in Figure 2.12 show the I/O values to and from the design in each tick the design was run for. The values that start as '0x' are input values, fed in hexadecimal format. Other values are the outputs which are in the decimal format. this is how WFGsim works, taking values in hexadecimalformat and outputting results in decimal.

Py2WFGsim can accept values in any base but it is also configured to mimic the behavior of WFGsim by default. It will look for this iov file upon startup, and if found, will simulate the design with those inputs and if unable to find this file, will generate a user configurable number of random input vectors (1000 by default) and run the simulation on them. User also has the option to define expected outputs for a given set of inputs and test it against the actual outputs. This makes Py2WFGsim a strong verification tool.

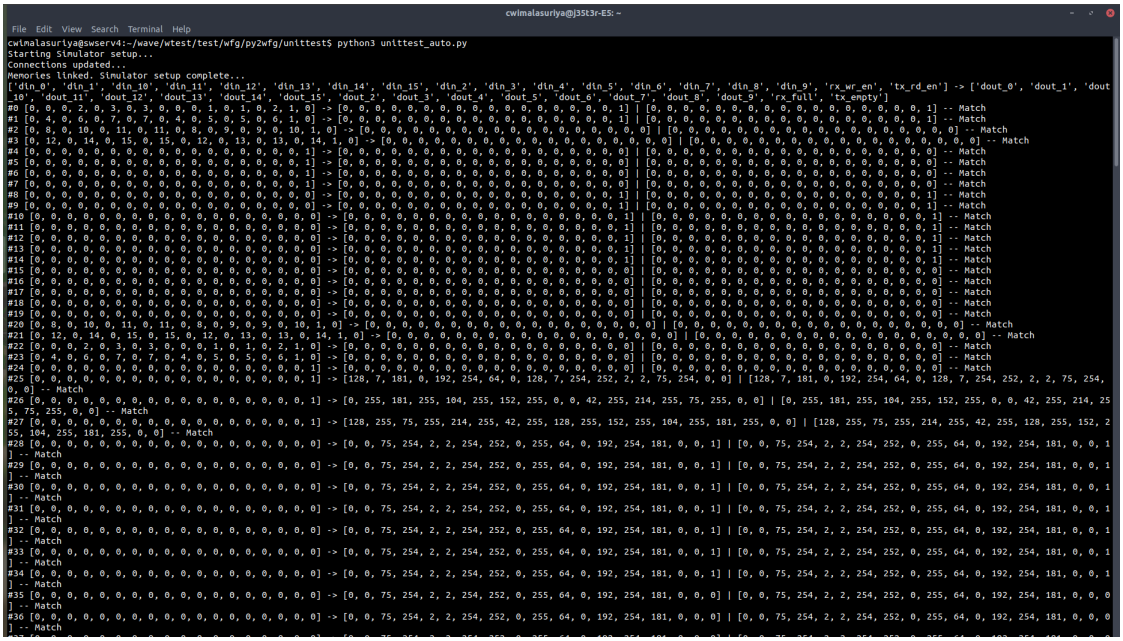


Fig. 2.13. Py2WFGsim user interface

2.4.5 DMA engine

The last part of the Project was implementing the DMA engine which was also the most controversial part since even now the engineers are having trouble with how this works. It is also known to slow down WFGsim and cause crashes and mismatches of all kind.

DMA controller needs to be capable of communicating with an external host and managing the data transfers within the chip. These transfers are asynchronous and can occur at any time, making it highly challenging to implement it in a software based platform. Therefore, we decided that the DMA transactions and processing of data do not need to be executed in parallel for the time being. Another important decision is to altogether eliminate multi-thread programming, because it was

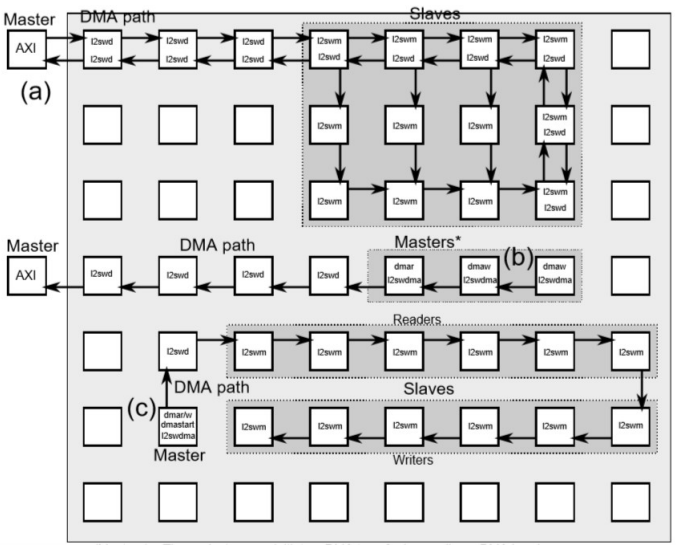


Fig. 2.14. Wave DPU DMA path

identified as one of the major culprits of the poor performance in the original WFGsim. We opted for a linear implementation where only one operation executed at one time.

The final implementation consisted of standalone hosts that user can create and link to the channels of the design. Then the host would have R/W queues for each channel which it would properly execute as the time in the simulator passes by. This allows the user to freely schedule the host behavior without considering complex timing constraints. This design was successfully integrated to the simulator and it got accepted by the SDK team as a capable tool.

2.4.6 Unit Testing

2.4.7 International Testing

To test out the translator/Simulator functionality, I was facilitated with a testing team consisting of people working from different countries including USA and China.

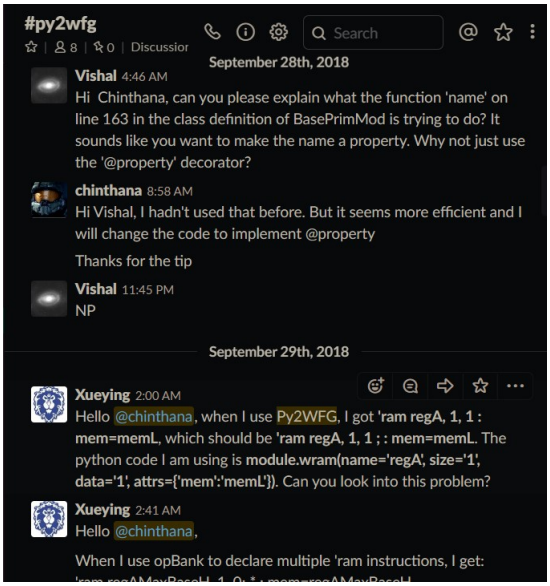


Fig. 2.15. Communications with the overseas testing team

This testing team was always helpful and efficiently reported whatever issues that the original designs had and came up with suggestions on best ways to solve the problems. They were a novel experience to me since I had never worked with a team scattered around the globe. It was rather efficient due to the software delivery system Wave has in place, allowing me to quickly deliver a fully tested builds of software to people in USA or China.

2.5 Wave external software tools

2.5.1 Overview

Working at Wave computing, I was exposed to the use of a lot of software automation tools that are used all over the world in design, development and testing of large scale industrial projects. Working with this expensive software with a crucial part of my internship experience.

The main external software tools used by wave are,

- GitLab
- Jenkins
- Confluence
- Jira
- Slack
- Zoom

Apart from these software wave computing uses other special software in testing procedures. In general view, working with these expensive software tools is one of the best experiences that I obtained during the internship of six months.

2.5.2 GitLab

Git is an open source protocol used to manage software code bases collaboratively. A collection of code stored on a git server is called a repository. Users can create 'clones' of the repository on their local computers and edit the code and push the changed code back into the repository. For easier management, users can also create branches in the local clones and make the edits in them, and then merge them back together, as shown in figure 2.16. Git protocol has been successfully adopted and rebranded by a number of companies incluing Github and GitLab.

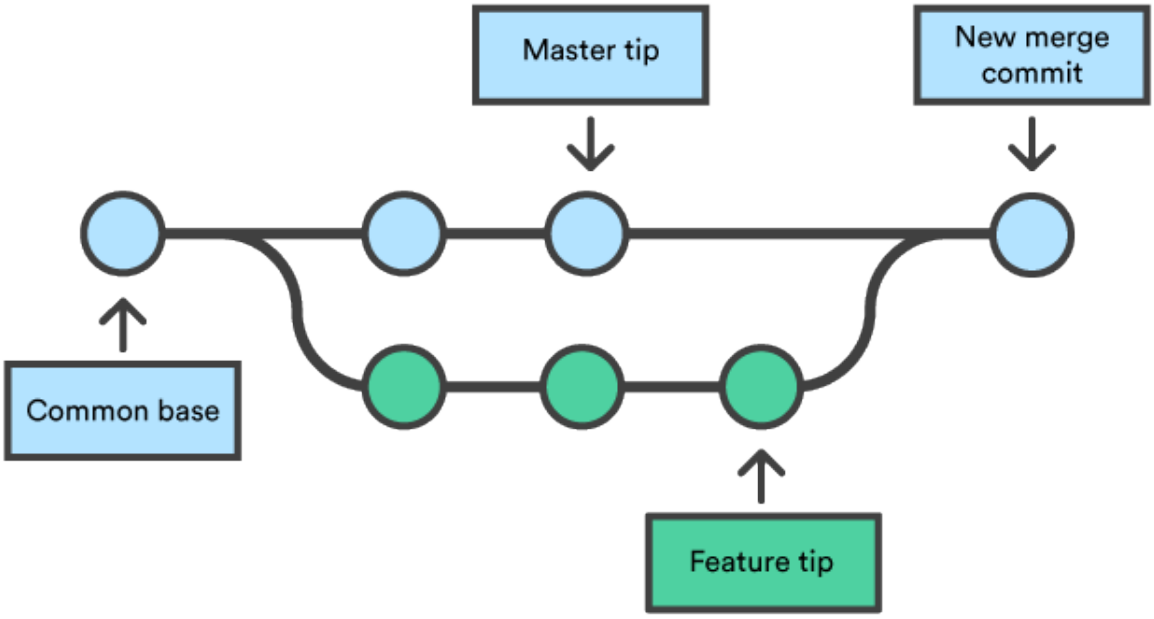


Fig. 2.16. Git protocol functionality

Wave computing uses GitLab as the main version control software. Since the use of version control in development of software project was not an entirely familiar topic for me I started working with them from scratch. I learnt about the efficient use of such systems and their various advantages in development of software projects. I realized the significance of such systems where developers from scattered everywhere could work according to a centralized plan and develop the same project from different operating from different parts of the world. Apart from that I experienced the advanced techniques used by large development communities in order to track every step of the development of a large scale project.

2.5.3 Jenkins

Wave computing uses Jenkins as the test base for their automated testing procedure. Use of Jenkins in testing was an entirely new experience for me. A test run on Jenkins covered up all tests that are needed to be run on any wave flow graph design. After compilation and verification

of the functionality of each wave flow graph design I was requested to run a mandatory test done on Jenkins before merging it with the main design of the master. In the event of failure of any of the designs I was assigned to investigate the reasons, rerun tests and make sure that the design was fully functional before engaging it with the main design.

2.5.4 Confluence

Documentation of all Wave Computing related details are maintained in a Confluence page. I witnessed the importance of documentation while exploring various archives of Wave design development tools. I realized the downside to apparent dissatisfaction among the developers in maintaining properly updated archives of software development projects. I ran into a number of troubles due to outdated or insufficiently developed documents, specially about the DMA engine of the WFGsim. This contained a number of conflicted documents scattered around the Confluence site. However, I managed to gather these documents and extract a final definition of the DMA engine and build a replacement on my own.

2.5.5 Jira

Jira is the official issue tracker for Wave Computing team. Any issues, bugs, improvements or changes to the wave design flow are recorded as tasks in Jira. Resolving these tasks can be assigned to any of the design individuals of the wave computing team. Usually the tasks come with supervisors and senior consultants in charge of completion of it. Any changes or developments to that is updated to the Jira so that everyone can access the information and get an idea about the current status of the Wave design tools. It is a very convenient way of managing projects and deciding on the development of software project carried out with collaborators from all over the world.

2.5.6 Slack

All the text based communication within Wave is carried out via Slack. It is like a normal instant messaging app that we have for our normal usage (Eg. Whatsapp) but has a lot of Enterprise focused features. It has the capability to separate work related content on channels, organize

notifications on priority and enhanced search/organization capabilities. It is also available in almost every platform, mobile, windows, linux or web, making it perfect for a company that uses many platforms on their computing systems.

2.5.7 Zoom

While slack handles texting, zoom is the video calling platform of Wave. As the management of the company is overseas, high quality video calling is essential for Wave. Zoom provides this with their highly optimized VoIP technologies. Even when the internet connection is poor, which is a common problem in Sri Lanka, Zoom can maintain an acceptable link over it.

2.5.8 Usage of tools in the SDK verification flow

The Wave SDK is a massive piece of software that has all the necessary features for writing a program for the DPU and testing it in a proper,thorough manner. This whole kit is still under construction and new edits need to be added to the code everyday but this should be done in a manner that does not break the existing code. This is done via a clever combination of GitLab and Jenkins as illustrated in Figure 2.17

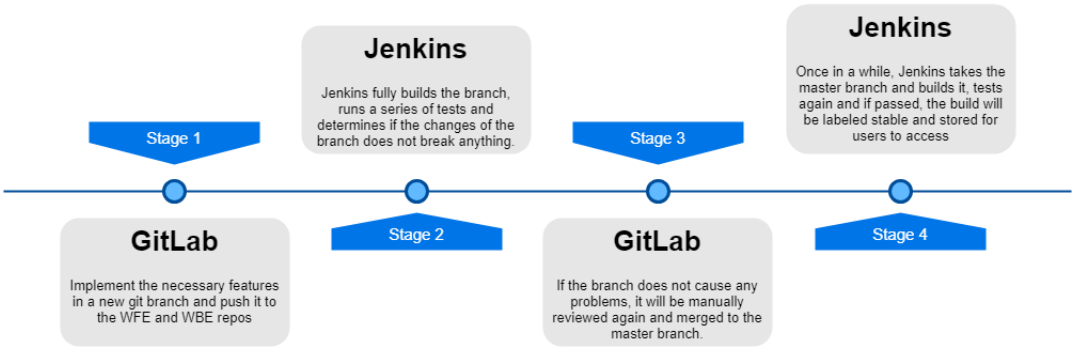


Fig. 2.17. Wave SDK build flow

2.6 Life at Wave Computing

Despite its multimillion dollar roots back in USA, Wave computing still adopts a startup culture that focuses on employee satisfaction ahead of profits. This combined with the fact that almost all the Wave employees are UoM ENTC graduates who are only a few years senior than ourselves make a very comfortable team to work with. Wave is a place where "Work while you work, Play while you play" is highly prominent. Whenever there is a Birthday, a wedding, a promotion or basically anything out of the ordinary becomes a reason to celebrate. Everybody knows Everybody very well and there is simply zero discrimination towards the interns. This place has one of the best communities I have worked with my entire life.

2.6.1 Daily routine

One of the notable things in Wave is that there is no practice of enforcing hours on people. Everyone is free to come and go as they wish but in the end, everyone has worked more hours than they would if there was any kind of enforcement. Everyone is happy to come to work and never once I have seen anyone procrastinating in the office.

Personally, I had selected my working hours to escape the traffic of the busy colombo city, coming to work before 7.30 in the morning and leaving at 4.00 in the evening. The day starts with initiating the toolchain build. the task is automated but takes a few minutes to complete. While the build completes, I check the communication channels for any contact from the HQ, since the time difference causes us to get an offset of around 14 hours. After checking communications and Jira task boards, The workload of the day can usually be easily planned. With lunch at around 1 pm, there is plenty of time to work. There were also a number of info sessions on how to use common tools such as C++, git, vivado, etc... If there is no planned work, I leave the office around 4 pm. And I'm very proud to say that I have not taken a single day of leave during the six month internship period.

2.6.2 Office facilities

Fig. 2.18. Typical Wave Computing Workstation

Wave Computing office provides its employees with state of the art working facilities with a comfortable working space, making it highly productive. There are conference rooms with Large displays that allow almost lifelike meetings with the HQ, there are cutting edge workstations (Figure 2.18) and basically everything you can expect from an office. It was absolutely a perfect place to work.

2.6.3 Supervisor Visits

About once in every 3 months one of the Senior officers from Wave USA comes to Sri Lanka for a direct supervision and review. While we were doing the internship, we had two such visits, both by our team supervisor Eng. Henrik Esbenson. These visits usually consists of individual feedback sessions, project reviews and brainstorm meetings. All of these are highly educational and worthwhile. In these meetings, we would get the chance to present our ideas on how our projects may progress and these ideas are highly respected and almost always accepted. It is customary to end the visit with a dinner in a high class restaurant where they celebrate the progress the company has made.

3 Conclusion

3.1 Training Summary

I worked as a training associate electronic engineer under Wave Computing division of Paraqum technologies, later separated as Wave Computing Sri Lanka. I worked on a single complex project that exposed me to all the aspects of the company.

The first part of the project was to develop a code translator to convert codes from python to Waves proprietary WFG format. This project took around two months to complete and it got accepted by the wave engineers. Second part, the simulator was a little more complex and it took around three months to fully build and test. It was also accepted and became a hot topic among the Wave engineering community overseas. Final month was spent on building the DMA engine fo the simulator. At the end of six months, i had finished all the tasks assigned to me and handed over the project in a state that may be further developed by wave later.

Getting this project working was challenging on many sides. First, I had to get Familiar with the proprietary DPU technology, which has absolutely no information available online. All the documentation on it was a single pdf file published only for internal usage. After that, there were a number of commercial grade tools such as Jira and Gitlab that had to be learnt from scratch. Then I had to learn how to work with my worldwide team. And after the programs were built, it had to be thoroughly tested according to Wave QA standards before giving them to the test users. As challenging these tasks were, I managed to complete them with the help of the people of Wave computing. As the training ended, I believe that I left a positive impression in the minds of the Wave Computing staff.

3.2 Self analysis

The project pushed my programming skills to the edge and I learned a lot of new things about How to deliver a commercial grade software product. It also taught me soft skills that i could never learn from the university. I also learned to fit in with the corporate culture and work with people

who are decades senior than me. It also helped me realize my position relative to the industry standards and I have recognized the following strengths and weaknesses of myself.

Strengths

- Have creative solutions to problems
- Can follow a plan or Improvise if necessary
- Have a widespread technical knowledge
- Can work with a team efficiently
- Can understand existing implementations with minimal references

Weaknesses

- Coding speed can be better
- Focus on a single task can be improved further
- Need some more knowledge on specialized areas such as Machine learning

Measures to Overcome weaknesses

- To improve coding speed and focus on a task, I intend to participate online coding competitions and practice codingin my free time.
- I also intend to do some specialized online courses on vital subjects.
- There is a Yoga practice module in semester 6B which I expect to help in focus issues.

3.3 Verdict on training establishment

As described in section 2.6, Wave computing is an organization that prioritizes employee satisfaction, which is no different with interns. The training experience from the work perspective is

also perfect. They managed to expose us to the cutting edge of electronics industry while maintaining a very comfortable working experience. I cannot imagine a better way to conduct the industrial training program other than the way Wave Computing managed to pull it off.

3.4 Verdict on Industrial training program

One of the key requirements when applying for a job nowadays is work experience. The NAITA industrial training program can be described as a perfect method of gaining experience during the degree time itself. It presents students with ample opportunities to define their future at their will. Almost all companies are eager to provide their support for the students in this program and practically everyone gets a good outcome from it. Talking about myself, I can doubtlessly claim that I had an exceptional industrial training experience that was well beyond my expectations.

References

[1] About Paraqum Technologies. https://paraqum.com/about.

[2] About TensorFlow. https://www.tensorflow.org/?hl=hi.

[3] About Wave Computing. https://wavecomp.ai/company.

[4] LLVM. https://en.wikipedia.org/wiki/LLVM.

[5] MIPS bought by wave computing. https://www.electronicsweekly.com/news/business/mips-bought-wave-computing-2018-06/.

[6] Wave Computing named a top 25 AI solution provider. https://globenewswire.com/news-release/2017/07/20/1054678/0/en/Wave-Computing-Named-a-Top-25-AI-Solution-Provider-for-2017.html.

[7] Wave computing raises 86m usd in oversubscribed series e round. https://wavecomp.ai/wave-computing-raises-86m-in-oversubscribed-series-e-round.