Framework for creating performance model of AI algorithms for early architecture exploration

Amit Dudeja, Amit Tara, Amit Garg and Tushar Jain

August 7, 2019

Framework for creating performance model of AI algorithms for early architecture exploration

Amit Dudeja, Senior Research & Development Engineer, Synopsys India Pvt. Ltd., Noida,India
(*adudeja@synopsys.com*)

Amit Tara, Senior Research & Development Engineer, Synopsys India Pvt. Ltd., Noida,India
(*amittara@synopsys.com*)

Amit Garg, Principal Engineer, Synopsys India Pvt. Ltd., Noida,India

(*amitgarg@synopsys.com*)

Tushar Jain, Research & Development Engineer, Synopsys India Pvt. Ltd., Noida,India
(*tjain@synopsys.com*)

Abstract— We are now living in an age of AI enabled smart homes, autonomous vehicles, and chatbots. Researchers are coming up with complex AI algorithms that are useful in various areas like speech and image recognition, even surpassing human accuracy. Increased complexity, growing competition and splurge in demand of AI enabled devices constantly put pressure on vendors to improve the chip performance and deliver the product as early as possible. System designers tend to use simulation techniques to quickly analyze and improve their model before implementing it in silicon. But till now these simulation techniques have not been leveraged for the AI domain. This paper proposes a novel framework to leverage Virtual Prototyping for early design exploration of power and performance targeting AI centric designs. We have used this framework to model a known Convolutional Neural Network, resnet-18 and simulated it on NVDLA hardware model. In our case study, we were able to determine key bottlenecks in the hardware design and were successful in our goal of reducing the inference latency to 5ms and minimizing the total energy and average power consumption.

Keywords—Deep Neural Networks, Virtual Prototyping, Design Space Exploration, Hardware Acceleration

I. INTRODUCTION

In the last few years, a lot of effort has been put to improve the vision-based algorithms and natural language processing. Researchers have been able to surpass human accuracy in tasks such as scene understanding, object detection, speech recognition. Deep Neural Network (DNN) has been the key driving force behind a broad spectrum of artificial intelligence applications. Unlike its predecessor, DNN automatically detect patterns and extract features from the training data. However, training these models have become a challenge as they require immense amount of data and computation power. Conventional CPUs have not been very helpful because of its inherent design while GPUs have fairly been successful in parallelizing the computation with its large number of processing cores and high data bandwidth thereby reducing the total inference time. Thus, in order to further improve the performance of a chip we require AI algorithm specific hardware accelerators to efficiently process DNN.

Depending upon the target platform, data center or edge, designing AI chips has forced companies to explore multiple factors such as power usage, memory bandwidth, data storage, area of the chip while having tradeoff of performance of AI algorithm and cost of a chip. Growing competition and desire to meet the requirements of AI algorithms has put a lot of pressure on companies to release new products in a short span of time with improved performance. Simulation techniques such as Virtual Prototyping (VP) has been widely used by the designers to efficiently model and deploy the architecture of a chip. Virtual Prototyping gives flexibility to a system architect to vary different parameters of a virtual hardware chip to get an optimal design in a short span of time. However, this technique has not been leveraged in designing hardware for AI algorithms.

The key contributions of this paper are as follows:

1. Translating AI algorithms into taskgraph based workload models. This is achieved by a novel framework which automatically generate taskgraphs from prototxt file.
2. Taskgraph based workload models are used for design space exploration of AI centric hardware.
3. A case study to present exploration of Resnet-18 (a CNN algorithm) running on NVDLA hardware model to achieve a target inference latency of 5 ms, with minimum power and energy consumption.

The rest of the article is organized as follows. Section II discusses the related work, Section III covers the framework and the Section IV covers the case study.

II. Framework Description

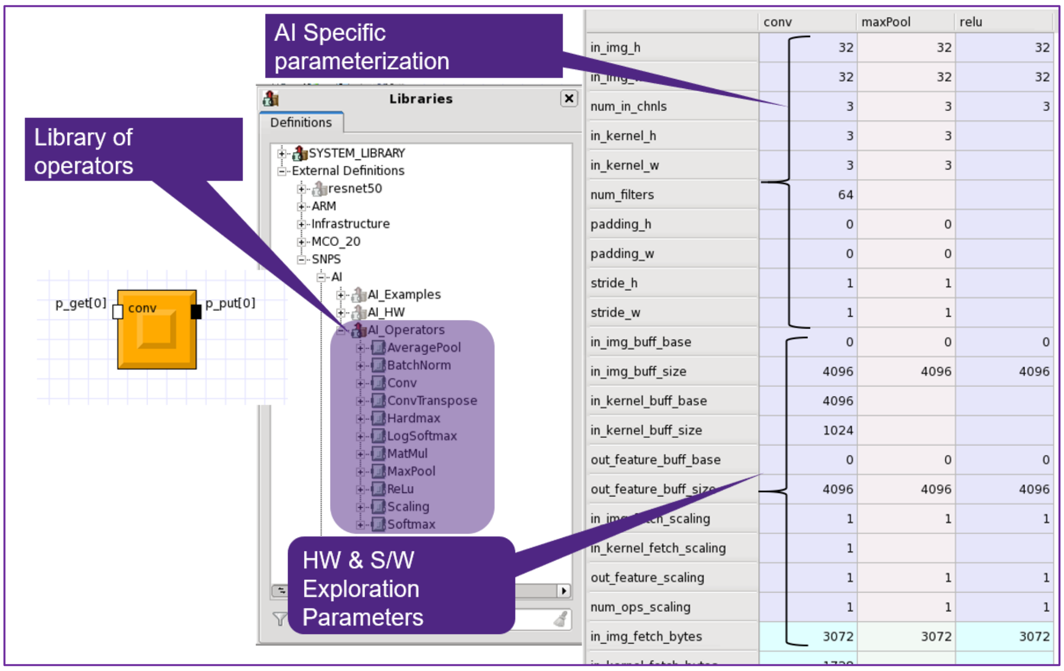
Early design exploration for AI centric hardware is a non-trivial task. It requires knowledge of both AI algorithms and hardware quite early in the design cycle. The lack of any comprehensive technique which can explore the AI algorithm in context of next generation AI hardware is the key motivation of this paper.
This paper proposes a framework:

- To create abstract baseline operators employed in AI algorithms
- To create performance model of AI algorithms from diverse set of inputs, using the abstract baseline operators.
- To create AI centric hardware, characterized with power and performance attributes.
- To explore AI algorithms in context of AI centric hardware.

The above aspects are briefly discussed as follows:

A. *Creation of abstract baseline operators employed in AI algorithm*

Baseline operators like Convolution, BatchNorm, AveragePool etc are the building blocks of any AI algorithm. In order to represent a performance model of an AI algorithm, it is required to first characterize baseline operators in terms of compute and memory load. Abstract baseline operator represents the load on the system in terms of processing cycles and read/write bytes and is not targeted for functional correctness.



	conv	maxPool	relu
in_img_h	32	32	32
in_img_[illegible]	32	32	32
num_in_chnls	3	3	3
in_kernel_h	3	3	
in_kernel_w	3	3	
num_filters	64		
padding_h	0	0	
padding_w	0	0	
stride_h	1	1	
stride_w	1	1	
in_img_buff_base	0	0	0
in_img_buff_size	4096	4096	4096
in_kernel_buff_base	4096		
in_kernel_buff_size	1024		
out_feature_buff_base	0	0	0
out_feature_buff_size	4096	4096	4096
in_img_fetch_scaling	1	1	1
in_kernel_fetch_scaling	1		
out_feature_scaling	1	1	1
num_ops_scaling	1	1	1
in_img_fetch_bytes	3072	3072	3072

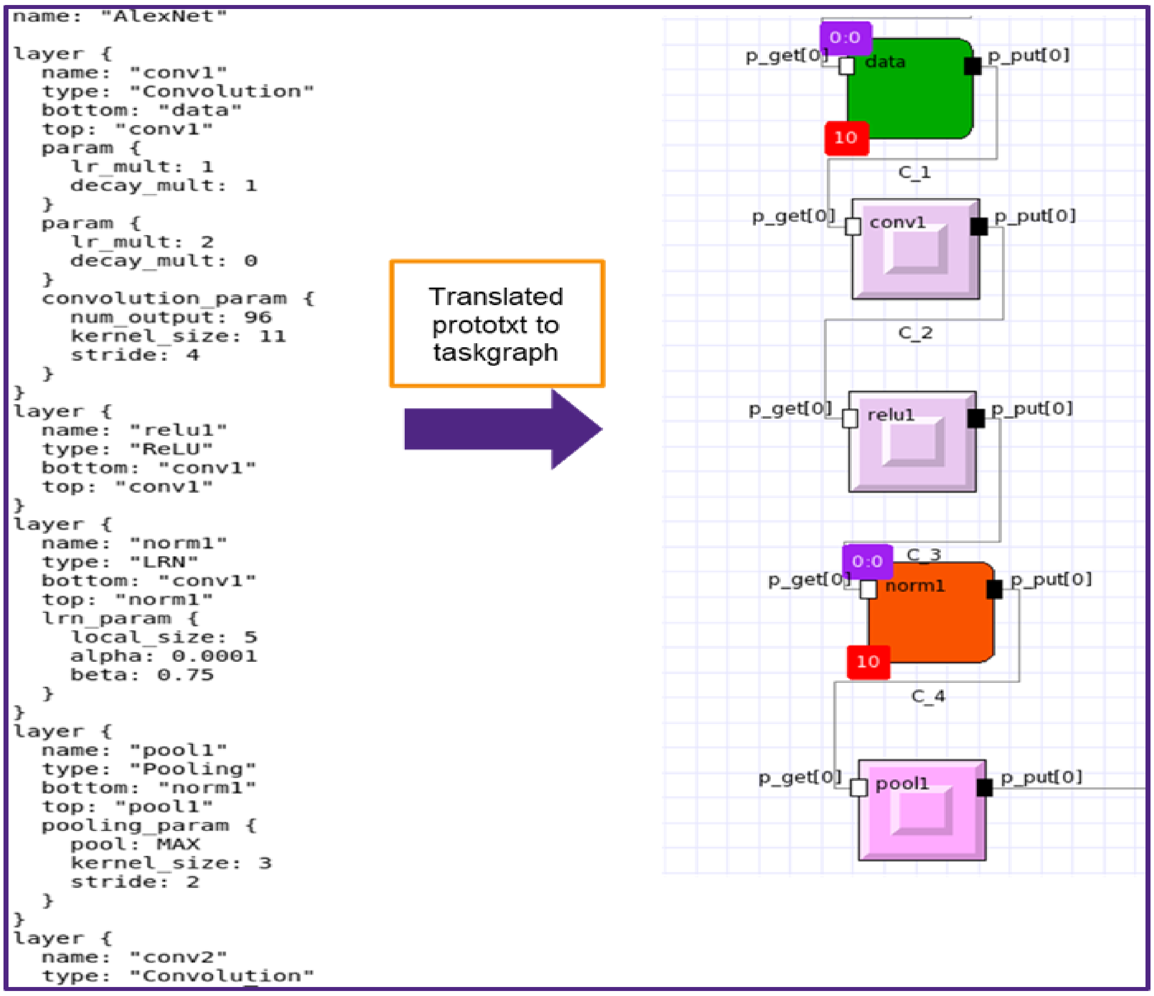
Fig 1: Abstract model of Convolution Operator created in Synopsys Platform Architect Ultra

Figure 1 shows Convolution operator, which is one of the most significant operators for any CNN algorithms. As depicted in the figure, the parameterization allows the operator to be configured for any set of inputs. Likewise, there are other essential operators in the AI Operators library.

B. *Creation of AI algorithm workload using baseline operators*

Creating taskgraph-based performance model of any AI algorithm can be a non-trivial task, as it requires an in-depth knowledge of the algorithm. This paper proposes a mechanism to automatically generate taskgraph from any AI algorithm such as resnet-18, vgg16 etc. This enables quick exploration of different algorithms for a hardware engine. With reduced turnaround time, a performance analyst can focus on optimizing the AI-centric hardware architecture.

AI workload model can be generated from a diverse type of inputs like software traces, profiler traces etc, however this paper focuses on creation of workload model from prototxt file format. Prototxt file is a text file that has structural information of a neural network in serialized fashion. It captures the complete topology of a neural network, where each layer is described with its attributes and connectivity with top and bottom layer.



```
name: "AlexNet"

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "norm1"
  type: "LRN"
  bottom: "conv1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "norm1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv2"
  type: "Convolution"
```

Figure 2: Snippet of the prototxt file on the left side and the translated workload in Synopsys Platform Architect Ultra on right hand side.

The framework provides a parser which automatically translates the algorithm, as captured in the prototxt file into taskgraph-based workload model. The parser scans the description of each layer, (in terms of its input interface, load attributes and output etc.), to create, connect and characterize an equivalent baseline operator. Once complete, a connected taskgraph is created (Figure 2).

AI Workload model can be checked in terms of its correctness and bottlenecks by simply simulating the workload model without any hardware design. To see the fidelity of the workload model, framework provides a mechanism to configure number of resources. Depending upon the number of configured resources, which by default is infinite, the elasticity of the workload model can be observed. It is also helpful in determining maximum theoretical gain an AI algorithm can achieve.

C. AI centric hardware engine

Due to popularity of AI algorithms in various domains, many hardware platforms have special features that target AI algorithm processing. For designing the next generation AI platforms, it is important to understand the compute and memory characteristics of an algorithm. This paper talks about a mechanism to model a configurable application-specific accelerator for improved throughput and energy efficiency.

For instance, a fundamental component in AI-algorithms, especially in neural networks, is multiply-and-add (MAC) operations, which can be easily parallelized. Techniques like vectorization (SIMD) and parallel threads (SIMT) can be highly instrumental in increasing the performance of the system. Furthermore, high parallel compute requires high memory bandwidth.

Here AI-centric hardware engine is modelled as Virtual Processing Unit (VPU) which is a composition of processing and memory resources. The process engine (PE) models the computation delay while the memory driver (MD) models the memory traffic in terms of read and write bytes. A VPU is configurable to have n-sub resources. It is also possible to construct hierarchies of heterogeneous or homogeneous VPUs to create a subsystem. To further characterize a VPU for performance studies, attributes like operations-per-cycle, stochastic cache, branch prediction, pipeline depth etc. can be enabled.

Systematically building a configurable hardware engine is of great importance while catering to different design strategies. This allows a performance analyst to capture families of designs with different trade-offs in capability, performance and energy efficiency. Such design flow for multiple architectures also facilitate re-use and comparison of configurable hardware engines. The AI algorithms are evolving very rapidly and to be ahead in competition the architecture must be optimized for a wider set of algorithms.

Synopsys Platform Architect Ultra provides a structured way to exercise various configurations for what-if analysis and optimization. The capability to record and visualize power & performance statistics across multiple simulations enables easy comparison of parameterized designs. Case study presented in the next section utilizes these features to optimize an AI SoC.

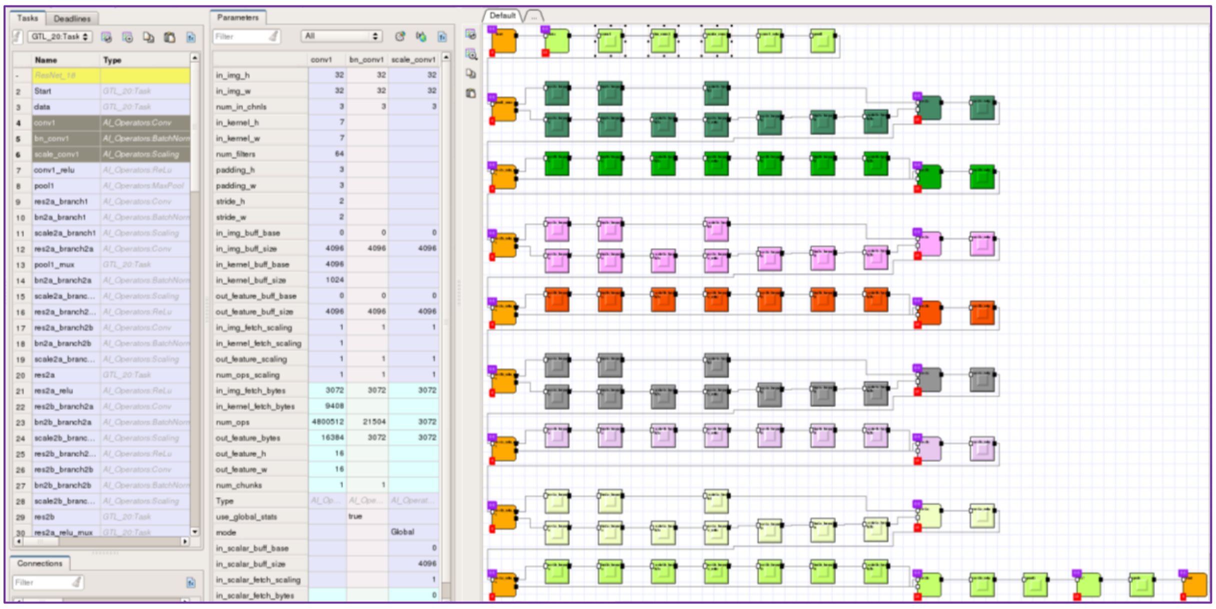
III. Case Study

This section covers a case study to demonstrate the AI Exploration framework for early architecture exploration. The study explores power and performance metrics of Resnet-18 algorithm executing on an NVDLA accelerator. The aim of this study is to achieve an inference latency of 5 ms as the key performance indicator (or KPI), along with power and energy as other optimization objectives.

A. WorkLoad Model

Resnet-18 [1] is a CNN model with 18 layers trained on ImageNet dataset comprising of 1 million images. The network can classify up to 1000 different objects and is a member of a family of CNN model which beat the state of art accuracy in image recognition (Imagenet large scale visual recognition challenge) in 2015. This family of CNN can be made deeper as it doesn't suffer from vanishing gradient problem and uses shortcut connection to improve the overall accuracy of the model.

Workload model of Resnet-18 is generated by parsing the prototxt file (Figure 3) and deducing the load characteristics in terms of compute and memory along with connectivity of each layer/operation. Synopsys Platform Architect Ultra tool automates the generation of workload model and provide plugins to further refine the model. For each operation/layer in the algorithm, there is a representative operator available in the tool. Operations like convolution, pooling etc can be easily modelled with these operators while creating the workload.



	Name	Type
-	ResNet_18	
2	Start	GTL_20:Task
3	data	GTL_20:Task
4	conv1	AI_Operators:Conv
5	bn_conv1	AI_Operators:BatchNorm
6	scale_conv1	AI_Operators:Scaling
7	conv1_relu	AI_Operators:ReLu
8	pool1	AI_Operators:MaxPool
9	res2a_branch1	AI_Operators:Conv
10	bn2a_branch1	AI_Operators:BatchNorm
11	scale2a_branch1	AI_Operators:Scaling
12	res2a_branch2a	AI_Operators:Conv
13	pool1_mux	GTL_20:Task
14	bn2a_branch2a	AI_Operators:BatchNorm
15	scale2a_branc...	AI_Operators:Scaling
16	res2a_branch2...	AI_Operators:ReLu
17	res2a_branch2b	AI_Operators:Conv
18	bn2a_branch2b	AI_Operators:BatchNorm
19	scale2a_branc...	AI_Operators:Scaling
20	res2a	GTL_20:Task
21	res2a_relu	AI_Operators:ReLu
22	res2b_branch2a	AI_Operators:Conv
23	bn2b_branch2a	AI_Operators:BatchNorm
24	scale2b_branc...	AI_Operators:Scaling
25	res2b_branch2...	AI_Operators:ReLu
26	res2b_branch2b	AI_Operators:Conv
27	bn2b_branch2b	AI_Operators:BatchNorm
28	scale2b_branc...	AI_Operators:Scaling
29	res2b	GTL_20:Task
30	res2a_relu_mux	GTL_20:Task

	conv1	bn_conv1	scale_conv1
in_img_h	32	32	32
in_img_w	32	32	32
num_in_chnls	3	3	3
in_kernel_h	7		
in_kernel_w	7		
num_filters	64		
padding_h	3		
padding_w	3		
stride_h	2		
stride_w	2		
in_img_buff_base	0	0	0
in_img_buff_size	4096	4096	4096
in_kernel_buff_base	4096		
in_kernel_buff_size	1024		
out_feature_buff_base	0	0	0
out_feature_buff_size	4096	4096	4096
in_img_fetch_scaling	1	1	1
in_kernel_fetch_scaling	1		
out_feature_scaling	1	1	1
num_ops_scaling	1	1	1
in_img_fetch_bytes	3072	3072	3072
in_kernel_fetch_bytes	9408		
num_ops	4800512	21504	3072
out_feature_bytes	16384	3072	3072
out_feature_h	16		
out_feature_w	16		
num_chunks	1	1	
Type	AI_Op...	AI_Ope...	AI_Operat...
use_global_stats		true	
mode			Global
in_scalar_buff_base			0
in_scalar_buff_size			4096
in_scalar_fetch_scaling			1
in_scalar_fetch_bytes			0

Figure 3: Resnet-18 performance model generated in Synopsys Platform Architect Ultra

B. *Hardware Model*

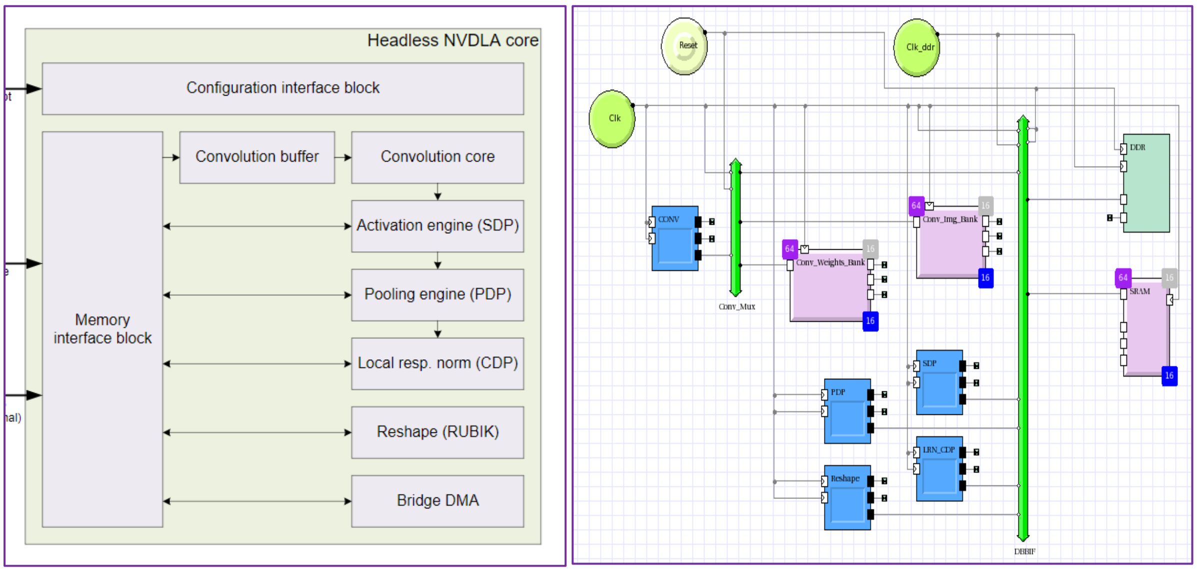


Figure 4: Real NVDLA core and equivalent Hardware model (blue boxes) created in Synopsys Platform Ultra

The Hardware model of the NVDLA in Platform Architect Ultra (Figure 4) represents the 5 main resources of the real NVDLA [5]:

- The Convolution core
- The Single Data Point Processor (SDP)
- The Planar Data Processor (PDP)
- The Cross-channel Data Processor (CDP) for local response normalization (LRN)
- The Data Reshape Engine

In this system, compute and DMA engines are modelled through VPU(s) and interconnects are represented by AXI bus. The design also contains a global SRAM and a global DDR memory controller.

Each operator is assigned a hardware resource to execute upon by mapping it to the appropriate VPU in the design. Initial run of a simulation, where Resnet-18 workload model is mapped to NVDLA accelerators with default configuration, gives a starting point to further analyze the system.

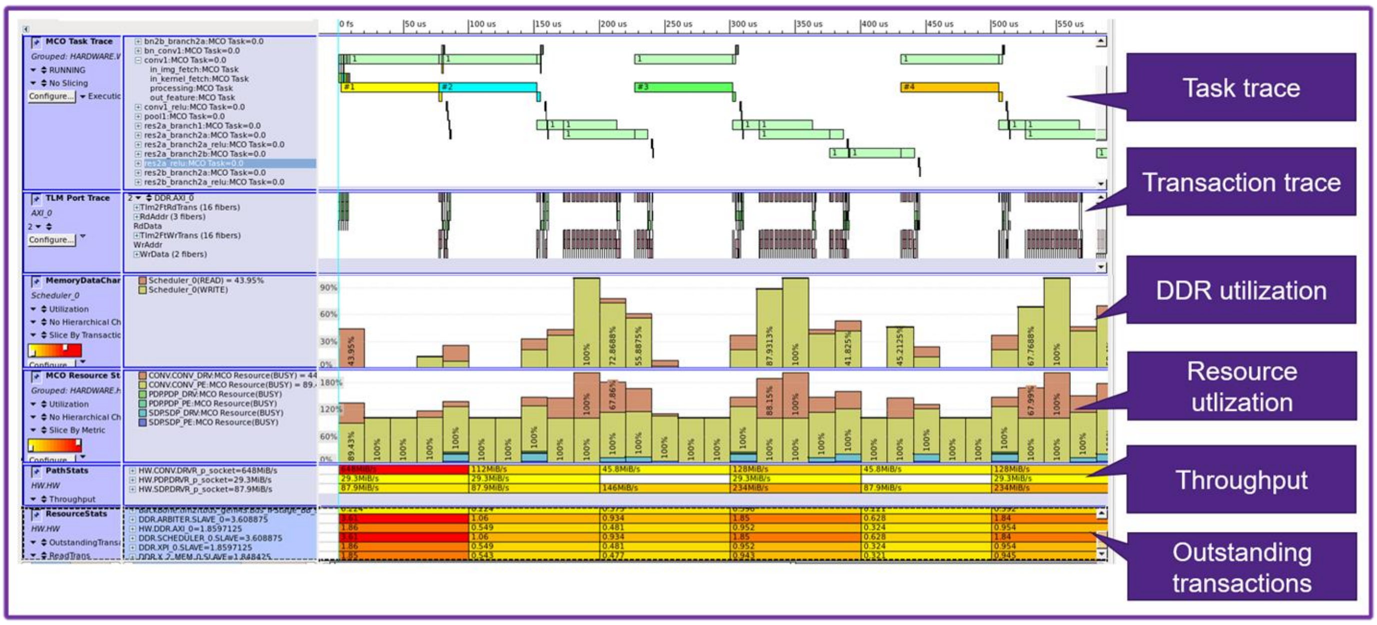


Figure 5: Different traces obtained with simulation executed with Initial Configuration

As seen in the figure above, the first trace depicts execution of a Resnet-18 taskgraph. The trace shows the activity of compute and memory centric tasks. Tasks *in_image_fetch*, *in_kernel_fetch* and *out_feature* represents the memory accesses while *processing* task represents the compute part of an operation. The prolonged activity of processing task shows that, in its default configuration, the performance of NVDLA is processing bound. This is also evident from the second trace which shows inactivity on memory ports for a long period of time. As a consequence, the DDR memory is under-utilized along with bus throughput and the number of outstanding transactions being low.

To accelerate the computations in the system, number of operations need to be parallelized by increasing the SIMD width. This can be easily exercised by setting up simulation sweep over parameters that configures the number of operations per cycle in the processing element.

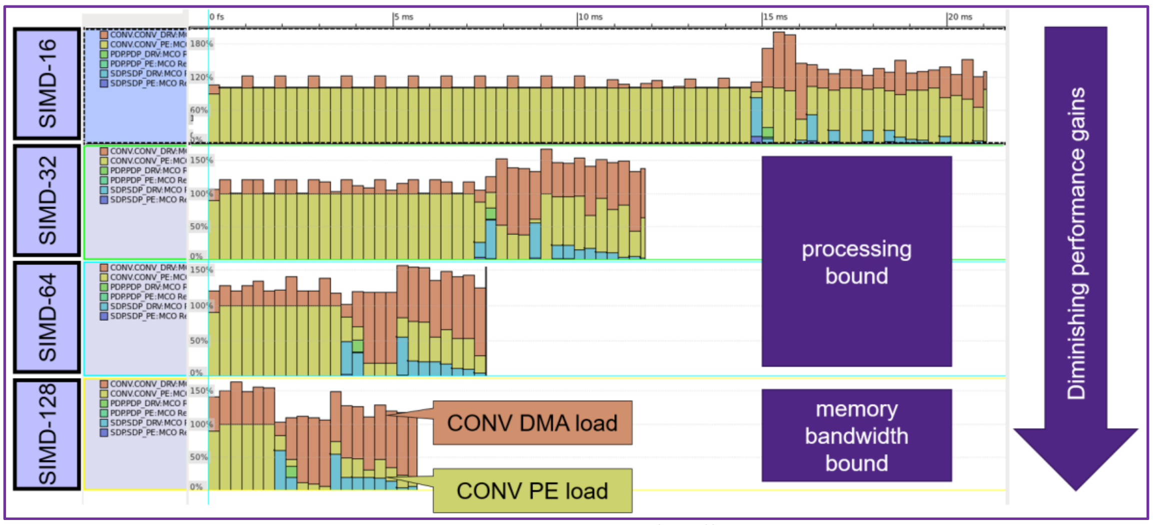


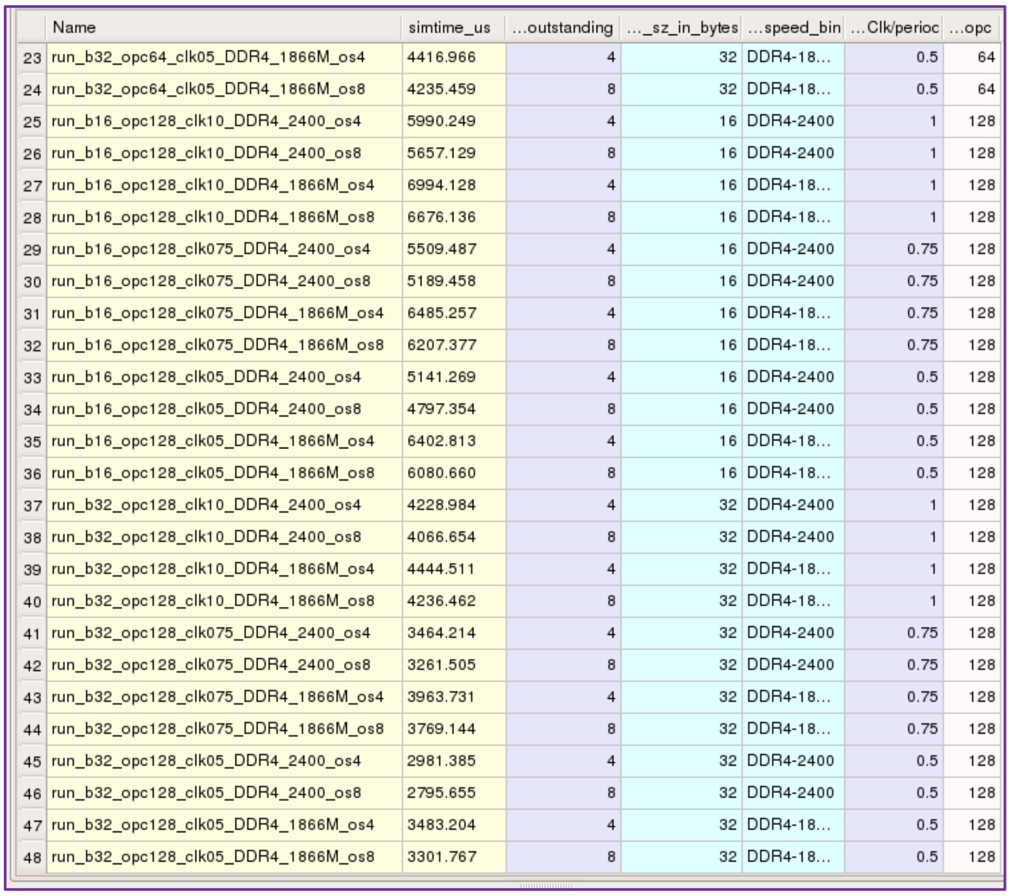
Figure 6: Resource utilization traces for different SIMD width

The result in Figure 6 shows the impact of SIMD width on the total execution time for ResNet-18 and the utilization of the DMA and the processing elements of the data path. As evident from the results, the performance is dominated by processing bound tasks until SIMD width is 32. With SIMD-64 and SIMD-128, the processing aspect seem to be reducing with memory bound activity dominating the performance. While doubling the SIMD width from 64 to 128, the performance gain looks to be diminishing. With this, it can be inferred that both SIMD-64 and SIMD-128 are promising data points. Our next steps shall be to optimize the memory path.

In the next step we refined the analysis by sweeping over different parameters in the hardware. Impact of the burst-size, the number of outstanding transactions, the DDR memory speed, the clock frequency and the SIMD width of the data path was studied. Following table shows the different sweeping parameters and the values that are tried to get the best scenario.

Sweeping Parameter	Sweeping Values
Burst size	16, 32
Outstanding transactions	4, 8
DDR memory speed	DDR4-1866, DDR4-2400
Clock frequency of data path	1, 1.33, 2GHz
SIMD width	64, 128 operations per cycle

Synopsys Platform Architect Ultra allows to sweep different parameters and run the simulations in parallel to extract the top-level power and performance metrics. Figure 7 shows the results of some of scenarios (primarily of 128 SIMD width) created from different values of sweeping parameters.



	Name	simtime_us	…outstanding	…_sz_in_bytes	…speed_bin	…Clk/perioc	…opc
23	run_b32_opc64_clk05_DDR4_1866M_os4	4416.966	4	32	DDR4-18…	0.5	64
24	run_b32_opc64_clk05_DDR4_1866M_os8	4235.459	8	32	DDR4-18…	0.5	64
25	run_b16_opc128_clk10_DDR4_2400_os4	5990.249	4	16	DDR4-2400	1	128
26	run_b16_opc128_clk10_DDR4_2400_os8	5657.129	8	16	DDR4-2400	1	128
27	run_b16_opc128_clk10_DDR4_1866M_os4	6994.128	4	16	DDR4-18…	1	128
28	run_b16_opc128_clk10_DDR4_1866M_os8	6676.136	8	16	DDR4-18…	1	128
29	run_b16_opc128_clk075_DDR4_2400_os4	5509.487	4	16	DDR4-2400	0.75	128
30	run_b16_opc128_clk075_DDR4_2400_os8	5189.458	8	16	DDR4-2400	0.75	128
31	run_b16_opc128_clk075_DDR4_1866M_os4	6485.257	4	16	DDR4-18…	0.75	128
32	run_b16_opc128_clk075_DDR4_1866M_os8	6207.377	8	16	DDR4-18…	0.75	128
33	run_b16_opc128_clk05_DDR4_2400_os4	5141.269	4	16	DDR4-2400	0.5	128
34	run_b16_opc128_clk05_DDR4_2400_os8	4797.354	8	16	DDR4-2400	0.5	128
35	run_b16_opc128_clk05_DDR4_1866M_os4	6402.813	4	16	DDR4-18…	0.5	128
36	run_b16_opc128_clk05_DDR4_1866M_os8	6080.660	8	16	DDR4-18…	0.5	128
37	run_b32_opc128_clk10_DDR4_2400_os4	4228.984	4	32	DDR4-2400	1	128
38	run_b32_opc128_clk10_DDR4_2400_os8	4066.654	8	32	DDR4-2400	1	128
39	run_b32_opc128_clk10_DDR4_1866M_os4	4444.511	4	32	DDR4-18…	1	128
40	run_b32_opc128_clk10_DDR4_1866M_os8	4236.462	8	32	DDR4-18…	1	128
41	run_b32_opc128_clk075_DDR4_2400_os4	3464.214	4	32	DDR4-2400	0.75	128
42	run_b32_opc128_clk075_DDR4_2400_os8	3261.505	8	32	DDR4-2400	0.75	128
43	run_b32_opc128_clk075_DDR4_1866M_os4	3963.731	4	32	DDR4-18…	0.75	128
44	run_b32_opc128_clk075_DDR4_1866M_os8	3769.144	8	32	DDR4-18…	0.75	128
45	run_b32_opc128_clk05_DDR4_2400_os4	2981.385	4	32	DDR4-2400	0.5	128
46	run_b32_opc128_clk05_DDR4_2400_os8	2795.655	8	32	DDR4-2400	0.5	128
47	run_b32_opc128_clk05_DDR4_1866M_os4	3483.204	4	32	DDR4-18…	0.5	128
48	run_b32_opc128_clk05_DDR4_1866M_os8	3301.767	8	32	DDR4-18…	0.5	128

Figure 7: Results summary of different simulations with different values of sweeping parameters

Pivot chart in Figure 8 shows the results from sweep of the different design parameters. The blue bars represent the total execution time for processing one frame of Resnet-18. Burst size seems to be most significant parameter, followed by the DDR speed, and the width and the clock frequency of the PE. The number of outstanding transactions has only a minor impact on performance (light vs. dark blue bars).

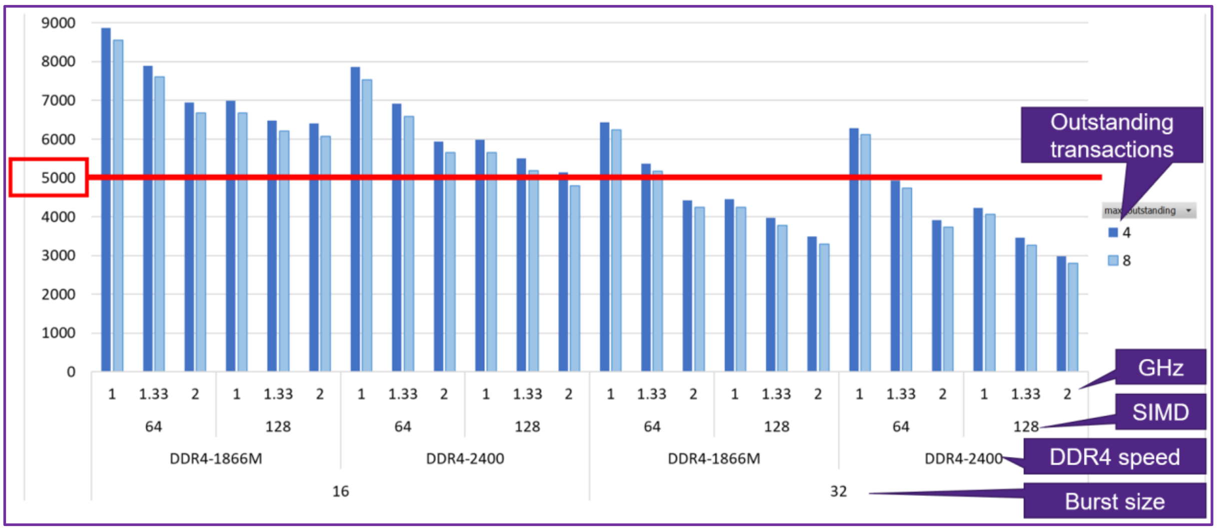


Figure 8: Pivot Chart to show results summary of different simulations with different values of sweeping parameters

The first step is to check, which configurations fulfill the KPI of 5ms interference latency. This limit is shown by the red line. For further analysis we only focused on the configurations, which satisfy the performance requirement.

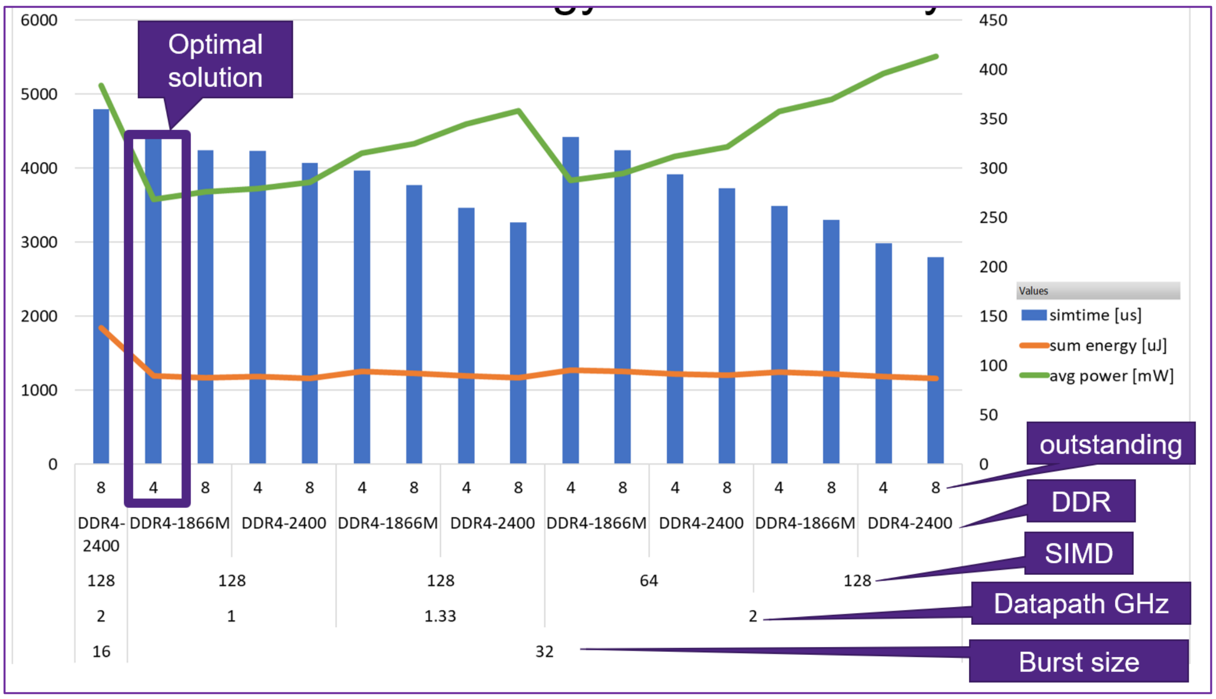


Figure 9: Filtered Pivot Chart to show configuration that meets the performance requirement

The above figure shows the filtered pivot chart for all configurations that meet the performance requirement. In addition to the blue bars for the total execution time the green line shows the average power dissipation in mW and the orange line shows the aggregated energy consumption in mJ. The power/energy consumption numbers have been achieved by mapping the UPF3.0 compliant power models on different blocks in Hardware.

The energy consumption is flat, so performance gain and increased power seem to level out over the different options. The 2GHz clock (results on right side) shows higher performance, but also drastically higher power consumption, especially for the wider data path. The 2nd configuration on the left shows the lowest power, with latency well below 5ms limit. Thus, the goal of 5ms latency with lower power and energy consumption is achieved with following configuration:

Hardware Parameter	Value
Burst size	32 bytes
Outstanding transactions	4
DDR memory speed	DDR4-1866
Clock frequency of data path	1GHz
SIMD width	128 operations per cycle

IV. CONCLUSION

In this paper we discussed the exploration of AI centric hardware engine, by automatically creating AI workload model from standard prototxt schema in conjunction with a characterized AI centric hardware engine. We also demonstrated a goal directed power and performance study to achieve an inference latency of 5ms with optimized power consumption.

V. REFERENCES

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[2] NVDLA Primer Documentation (http://nvdla.org/primer.html)
[3] IEEE (Std 1801-2015) Standard for Design and Verification of Low-Power, Energy Aware Electronic Systems
[4] Energy-Aware System Level Design using UPF3.0 (IEEE1801-2015), DVCON India 2016, Amit Dudeja
[5] IEEE (Std 1666-2011) Standard for Standard SystemC® Language Reference Manual