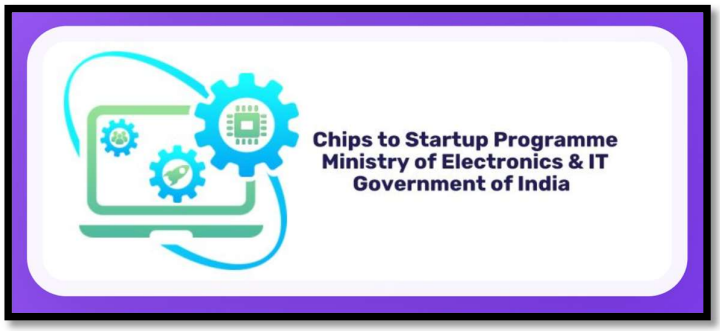


Real-Time Silicon Wafer Defect Image Classification Using a Hardware–Software Co-Designed MobileNetV2 Accelerator on a Xilinx Zynq SoC

Team SILLYCON
Rhythm Patel
Priyanshu Tyagi
Dr. Sparsh Mittal (Mentor)

ABSTRACT

In semiconductor wafer manufacturing, defect detection and classification are critical processes that directly influence production yield, cost efficiency, and time to market. The increasing resolution of wafer maps and the demand for low-latency, on-site inspection challenge conventional CPU-based inference systems, which suffer from limited throughput and higher energy consumption on embedded platforms.

This work presents a real-time silicon wafer defect classification system implemented using a hardware–software co-designed MobileNetV2 accelerator on a Xilinx Zynq SoC. The heterogeneous architecture leverages the Arm processing system (PS) for image acquisition, preprocessing, control logic, and post-processing, while compute-intensive convolutional operations are offloaded to the FPGA programmable logic (PL). The CNN accelerator is implemented using RTL with optimized pipelining, parallel multiply–accumulate (MAC) units, and on-chip BRAM buffering to minimize memory bottlenecks and maximize data reuse.

Experimental evaluation demonstrates significant performance improvement compared to a CPU-only implementation on the Arm processor, achieving real-time inference with enhanced throughput and reduced latency while efficiently utilizing FPGA resources (LUTs, DSPs, and BRAM). The proposed system validates the effectiveness of hardware-software co-design for deploying energy-efficient edge AI solutions in semiconductor inspection environments.

Introduction and Research Background

Semiconductor wafer inspection is essential for maintaining yield and reliability in IC fabrication, but as defect patterns become more complex with technology scaling, traditional inspection methods become computationally expensive and unsuitable for real-time deployment. While convolutional neural networks (CNNs) improve defect classification through automated feature extraction, they require significant computational and memory resources. Running them on embedded CPUs often fails to meet real-time and energy constraints, and cloud-based solutions introduce latency, cost, and privacy concerns, limiting their practicality for inline fab-floor applications.

To address these challenges, this work proposes a hardware–software co-designed inference system implemented on a Zynq-7000 platform from Xilinx. The Zynq architecture integrates a dual-core Arm Cortex-A processing system (PS) with FPGA programmable logic (PL) on a single chip, enabling heterogeneous acceleration through efficient workload partitioning. In the proposed architecture, image acquisition, preprocessing, system control, and post-processing are executed on the Arm processing system, while computationally intensive CNN layers are offloaded to the programmable logic for parallel hardware execution.

The selected backbone network is MobileNetV2, due to its lightweight structure and efficiency-oriented design. MobileNetV2 employs depthwise separable convolutions and inverted residual blocks to significantly reduce multiply–accumulate (MAC) operations and parameter count compared to conventional convolutional networks. This architectural efficiency makes it well-suited for FPGA-based implementation under constrained logic, DSP, and on-chip memory resources.

A baseline software-only implementation of MobileNetV2 was first executed on the Arm Cortex-A processor to establish reference performance metrics. Profiling analysis confirmed that convolutional layers dominate execution time and memory bandwidth utilization, accounting for the majority of inference latency. Based on this observation, a custom FPGA accelerator was developed to target the primary computational kernels of the network : Standard Convolution, Depthwise Convolution and Pointwise Convolution.

The accelerator architecture incorporates parallel MAC arrays, pipelined datapaths, and on-chip BRAM buffers to maximize data reuse and minimize off-chip memory accesses. Communication between the PS and PL is implemented using AXI-based interfaces: AXI4-Lite for configuration and control, and AXI4-Full or AXI DMA for high-throughput data transfer. A centralized finite state machine (FSM) coordinates execution sequencing and synchronization between software and hardware components.

Compared to traditional machine learning approaches such as Support Vector Machines and Random Forests—which rely on handcrafted features and limited scalability the proposed CNN-based accelerator enables automated feature extraction with improved classification robustness. Additionally, unlike computationally intensive architectures such as ResNet or Vision Transformers, MobileNetV2 provides a favorable trade-off between accuracy and computational complexity, making it practical for real-time deployment on heterogeneous embedded SoCs.

By leveraging the hardware–software co-design capabilities of the Zynq platform, this work demonstrates a scalable and energy-efficient edge AI solution for silicon wafer defect classification. The proposed system achieves measurable latency reduction compared to CPU-only execution while maintaining efficient FPGA resource utilization. This implementation validates the effectiveness of Arm–FPGA heterogeneous architectures for industrial edge AI deployment and establishes a foundation for future enhancements such as precision quantization, sparsity-aware acceleration, and extended defect localization capabilities.

Functional Specifications and Methodology

1. Input

- High-resolution wafer images acquired from inspection equipment.
- Images are preprocessed and quantized to INT8 format to match the fixed-point FPGA accelerator architecture.

2. Model Architecture

- MobileNetV2 CNN quantized to INT8 using Quantization-Aware Training (QAT) to reduce computational and memory overhead while maintaining accuracy.
- Supported layers include Standard Convolution (SC), Depthwise Convolution (DW), and Pointwise Convolution (PW), along with Batch Normalization, ReLU, Max Pooling, and Fully Connected layers.
- Convolution and pooling operations are mapped onto parallel MAC/MAX-based Processing Elements (PEs) implemented in the programmable logic.

3. Output

- Predicted wafer defect class is transmitted to the Arm Cortex-A Processing System (PS) via AXI interface for further decision-making and system-level control.

4. Performance Metrics

- **Accuracy:** INT8 quantized model achieves ~95.5% classification accuracy compared to ~97.8% FP32 baseline.
- **Latency:** ~3.92 ms per image using fully pipelined hardware execution.
- **Throughput:** ~255 images per second, enabling real-time inspection.
- **Energy Efficiency:** Reduced power consumption through fixed-point arithmetic and pipeline-based architecture.

5. Memory Architecture

- On-chip BRAM buffers store feature maps, weights, biases, scales, and shifts for each CNN layer. Data movement (inputs, intermediate activations, outputs) is managed through AXI4 interfaces with burst transfer support.
- DDR memory can be used for storing larger model parameters and input datasets when required.

6. Interfaces and Integration

- AXI4-Lite for control and configuration (start/stop, status, layer parameters).
- AXI4/AXI-DMA for high-speed data transfer between PS and PL.
- Fully memory-mapped accelerator tightly coupled with the Arm processor within the Zynq SoC environment.

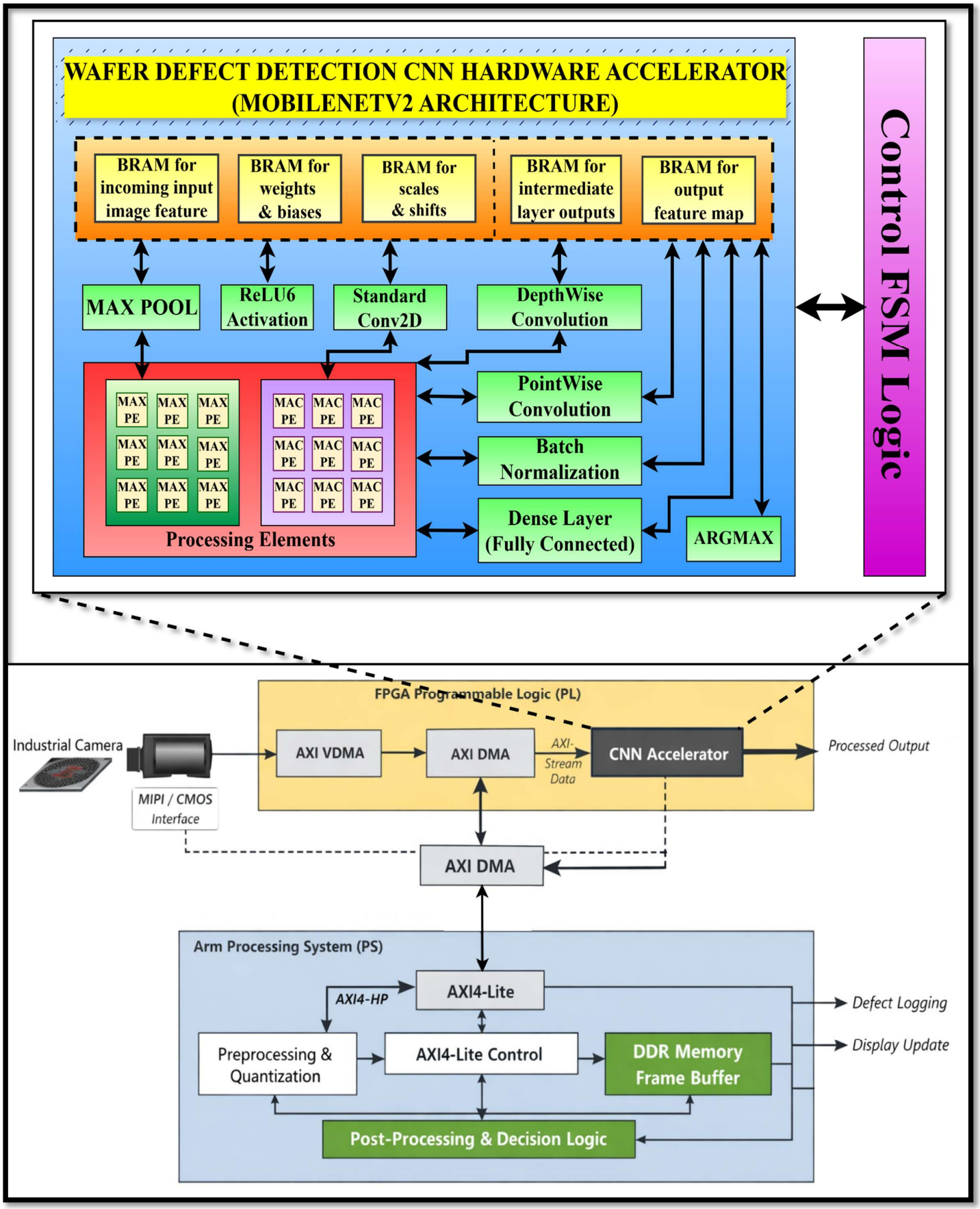
7. Quantization and Pipeline Optimization

- MobileNetV2 converted from FP32 to INT8 using QAT, reducing computational complexity by ~75% while preserving accuracy.
- Intra-layer and inter-layer pipelining maximize PE utilization and minimize inference latency.

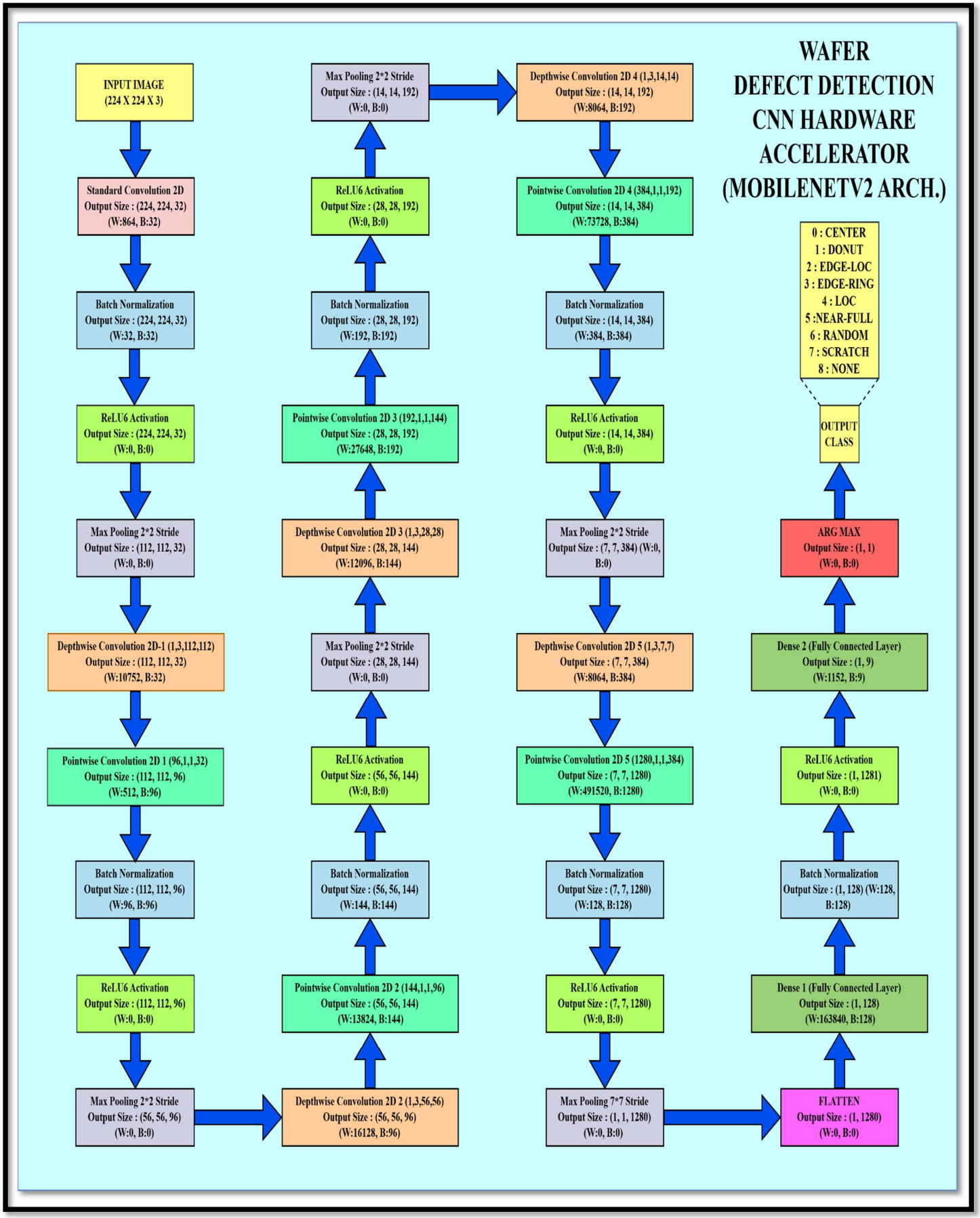
8. Tools and Methodology

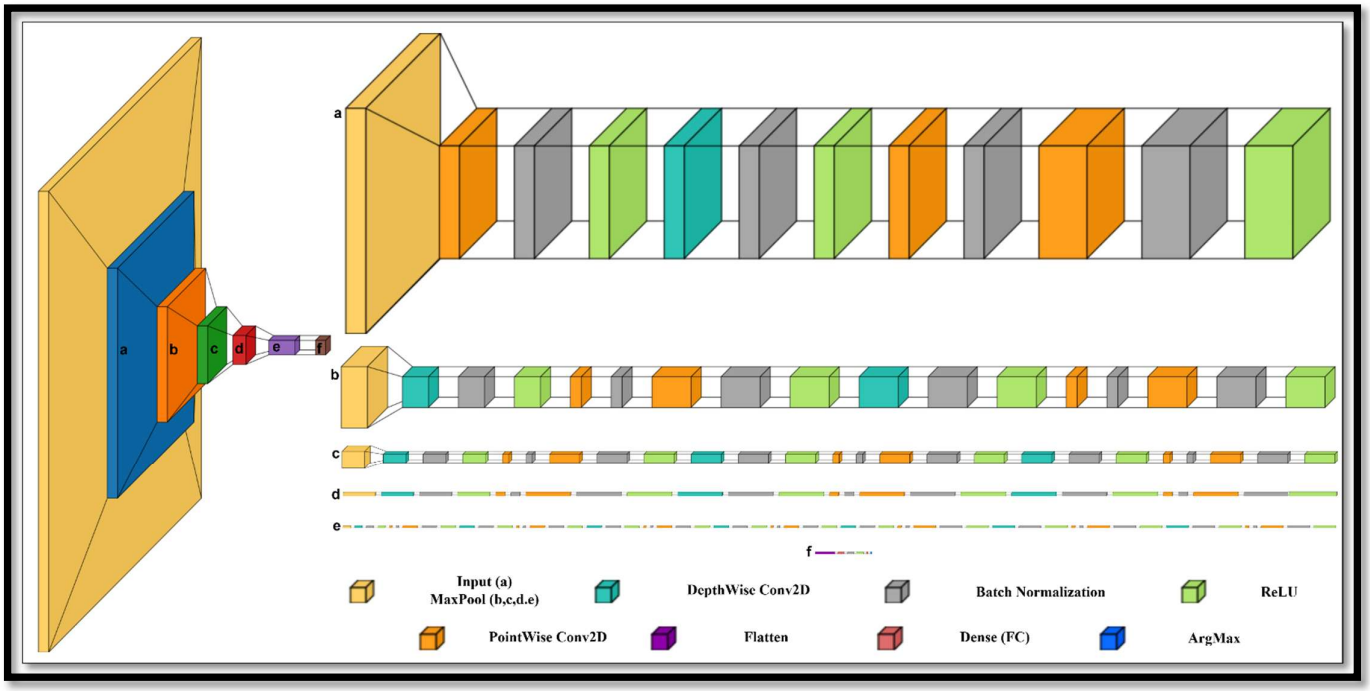
- **Design & Synthesis:** Xilinx Vivado for RTL synthesis, implementation, and bitstream generation.
- **Hardware Description:** Verilog/SystemVerilog for accelerator design.
- **Simulation & Verification:** Vivado Simulator for functional and timing validation.
- **Model Training & Quantization:** Python with TensorFlow(QAT) for training and INT8 conversion.

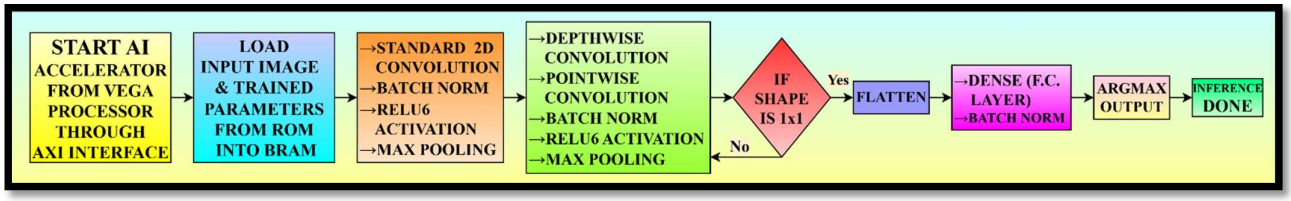
High-Level Architecture of the Proposed Design



Layer-wise Design Flow of AI Hardware Accelerator







Input

- Image size: 224 × 224 × 3 (RGB wafer image).
- This serves as the raw input vector for the CNN accelerator.

Standard Convolution (3×3, same padding)

- Operation: Applies 32 filters of size 3×3 across the input image with padding to preserve dimensions.
- Output size: 224 × 224 × 32.
- Processing elements: Each filter is implemented using multiple MAC PEs (Multiply-Accumulate Processing Elements) to perform convolution in parallel.
- Purpose: Extracts low-level spatial features such as edges, scratches, and texture anomalies from wafer images.
- Trained parameters: Weights + biases.

Batch Normalization

- Operation: Normalizes feature maps across each channel.
- Parameters: Scale (γ) and shift (β), both learned during training.
- Purpose: Stabilizes training, improves convergence, and prepares data for quantization.
- Output: Same shape, normalized values.

ReLU6 Activation

- Operation: Activation function with clipping at 6 → maps values to [0, 6].
- Significance: When quantized to INT8, values fit into 4 bits, improving power efficiency.
- Purpose: Introduces non-linearity while controlling dynamic range.

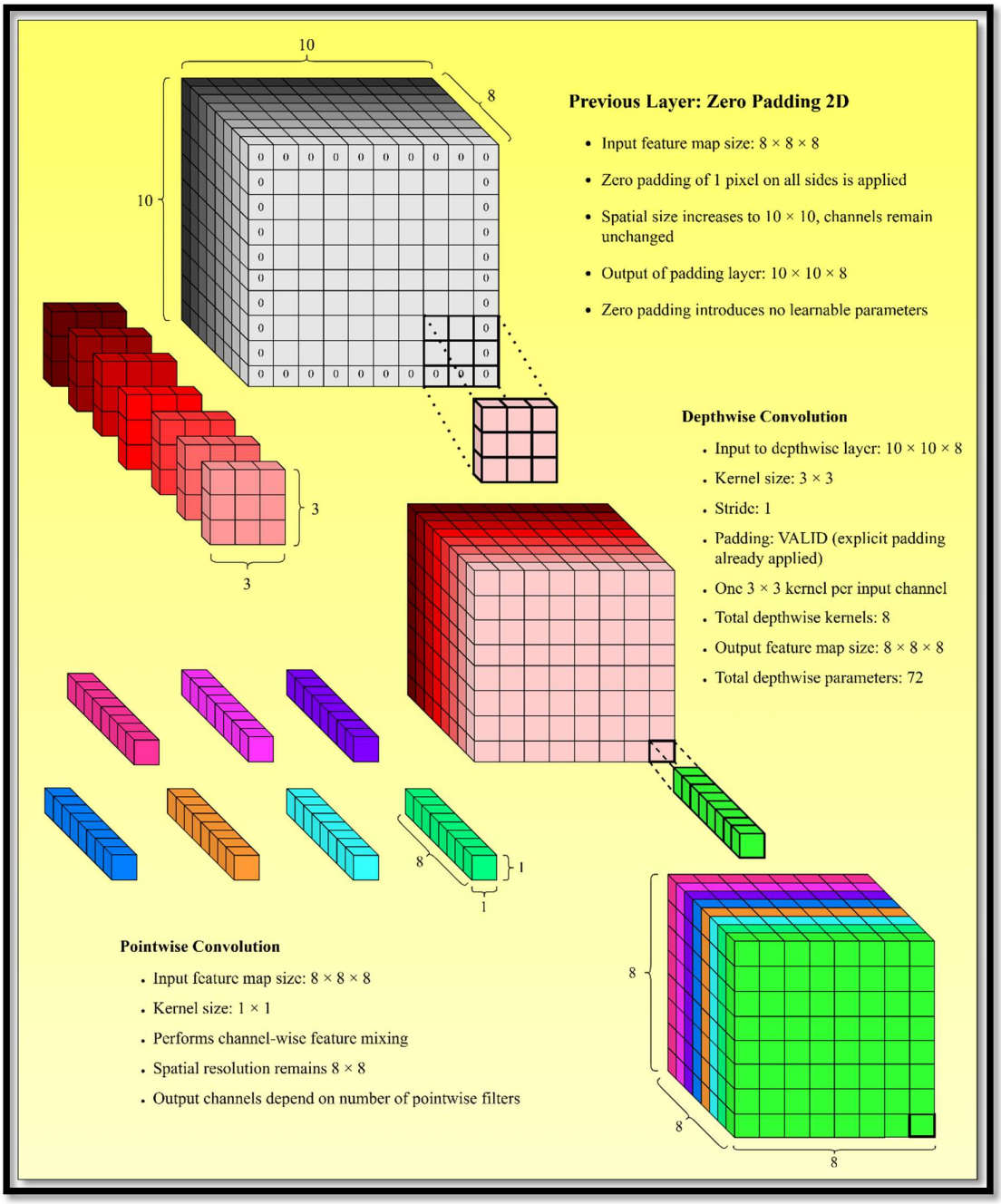
Max Pooling (2×2 stride)

- Operation: Picks maximum value from every 2×2 region.

- Processing element: Max PE compares 4 inputs and outputs the largest.
- Output: Reduced spatial size (downsampling) while retaining strongest features.
- Purpose: Reduces computation, increases robustness to translation.

Depthwise Convolution (3×3, zero padding)
- Operation: Each input channel has its own filter (channel-wise convolution).
- MAC usage: Much fewer MACs than standard conv, since no mixing across channels.
- Purpose: Lightweight feature extraction, preserves channel independence.
- Trained parameters: Depthwise filter weights + biases.

Pointwise Convolution (1×1)
- Operation: Combines depthwise outputs across channels using 1×1 convolutions.
- MAC usage: Intensive, but highly parallelizable via many PEs.
- Purpose: Channel mixing and dimensionality adjustment.
- Trained parameters: Weights + biases.



Iterative Blocks

The pipeline alternates [Depthwise → Pointwise → BN → ReLU6 → Pooling] multiple times:

- Each block extracts higher-level features : From scratches and edges to larger defect structures and Reduces spatial resolution while increasing depth (channels).
- Max pooling in early stages = 2×2 stride.
- Final pooling = 7×7 stride → produces a compact global representation.

Flattening Layer

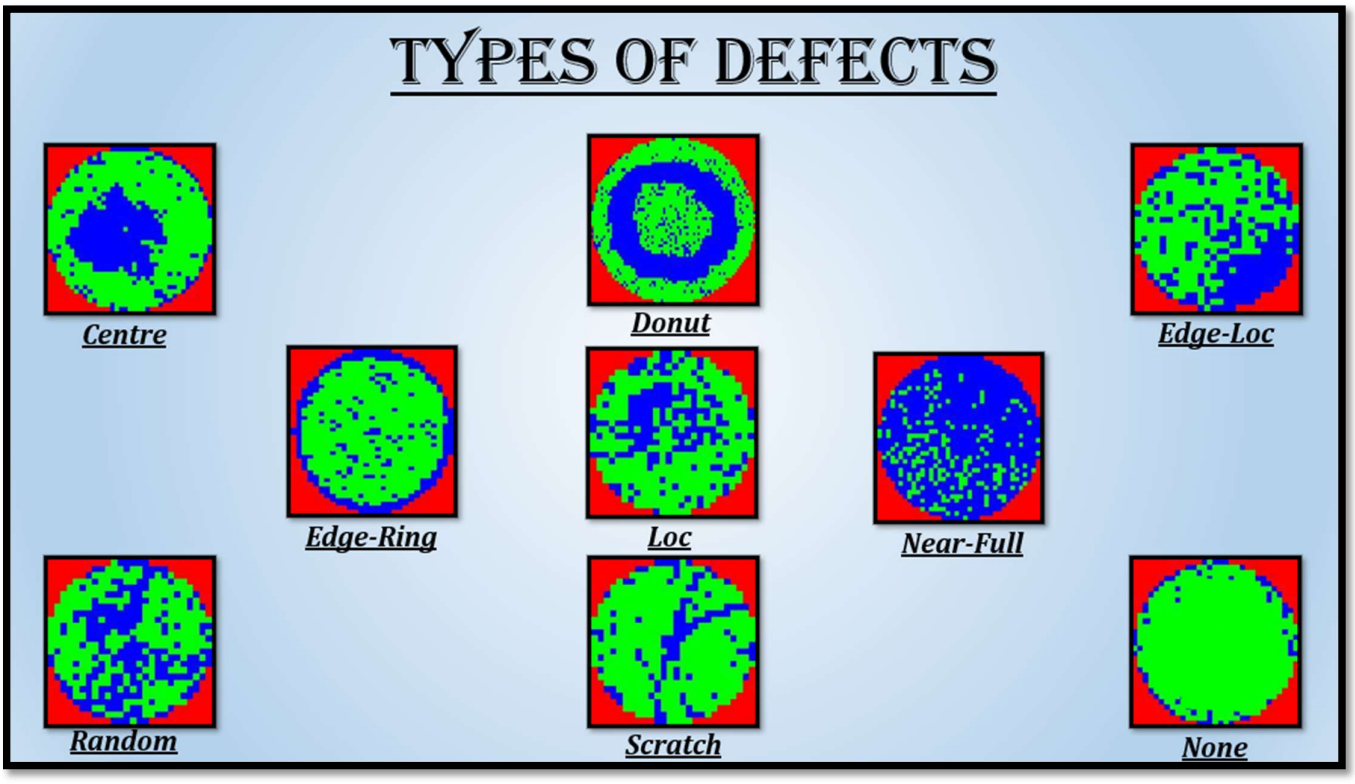
- Converts the final 7×7×1280 tensor into a 1D vector.
- This vector is the condensed "signature" of the wafer image.

Dense (Fully Connected) Layers

- Dense 1: Connects flattened vector to hidden representation (all MAC operations).
- Dense 2 (final): Maps to 9 output neurons (classes).
- Parameters: Large number of MAC operations since every neuron connects to all inputs. Weights and biases are learned.

ArgMax

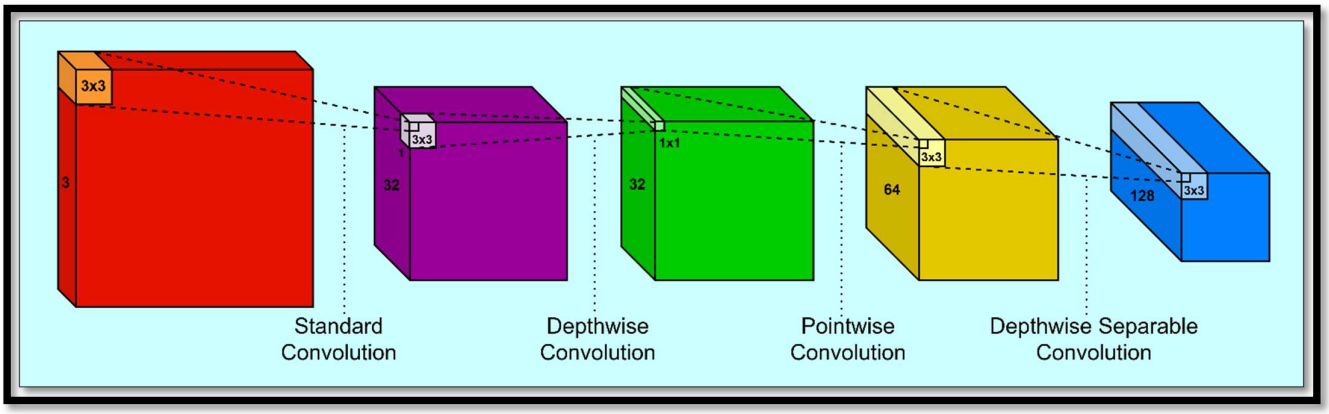
- Operation: Selects the index of the maximum logit among the 9 classes.
  Classes:
  0 : Center,
  1 : Donut,
  2 : Edge-Loc,
  3 : Edge-Ring,
  4 : Loc,
  5 : Near-Full,
  6 : Random,
  7 : Scratch and
  8 : None
- Output: Final predicted wafer defect category.



Significance of Pipelining (Optimization Techniques)

In the proposed accelerator, each computational stage of MobileNetV2 is carefully mapped to hardware primitives to ensure maximum efficiency under edge deployment constraints. The pipeline is structured as follows:

- **Convolutions (Standard, Depthwise, Pointwise):** These layers perform hierarchical feature extraction from wafer images. Each convolutional block is mapped to a bank of optimized MAC-based Processing Elements (Pes), allowing parallel computation of kernel operations.
- **Batch Normalization + ReLU6:** These operations stabilize training and inference while enabling INT8 quantization. The ReLU6 activation, in particular, bounds activations to 6, making it well-suited for FPGA-based fixed-point deployment.
- **Pooling:** Max pooling layers progressively reduce spatial dimensions while preserving salient features. Earlier pooling stages (2×2) downsample intermediate feature maps, whereas the final 7×7 global pooling aggregates contextual information across the entire wafer image.
- **Dense Layers:** Fully connected layers integrate extracted features for high-level decision-making. Each neuron is realized as an array of MAC PEs performing multiply-accumulate operations across the flattened input vector.



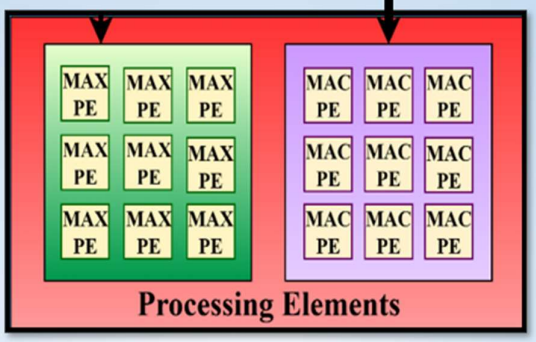
To illustrate the efficiency of our pipelined hardware design, consider the Depthwise Convolution layer with an input feature map of size 32×32×32 and a kernel size of 3×3. This computation requires approximately 294,912 MAC operations. In proposed architecture, MAC PEs are organized into a parallel 9-PE cluster, where each PE computes one kernel window per cycle.

Through intra-layer pipelining, the workload is distributed such that while one PE computes the first receptive field, the others simultaneously compute adjacent windows. This reduces the effective latency per feature map row from 288 cycles (sequential) to 32 cycles (parallelized).

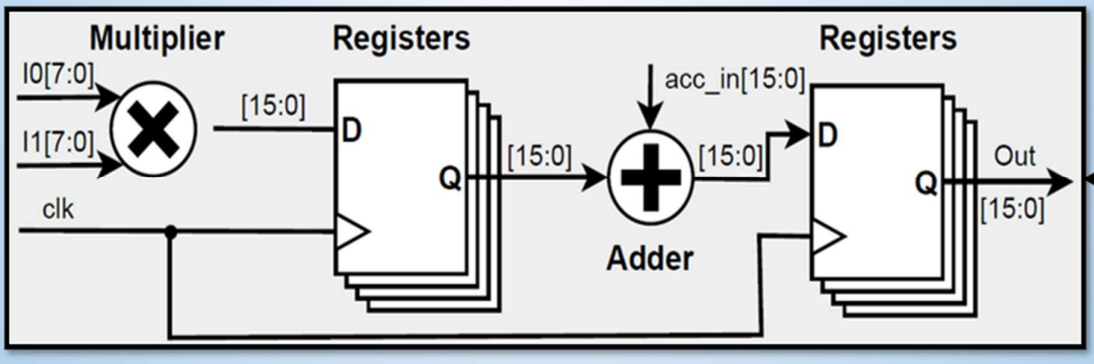
In addition, inter-layer pipelining is employed to overlap computations between consecutive layers. For instance, while DWCE produces its output feature maps, the subsequent Pointwise Convolution immediately begins consuming them without waiting for full completion. This overlap hides data dependencies across layers and maximizes throughput.

Post-synthesis results on Lattice Radiant confirm that this combined pipelining strategy yields a 3.1× reduction in inference latency compared to a non-pipelined baseline, achieving PE utilization efficiency while maintaining classification accuracy close to FP32.

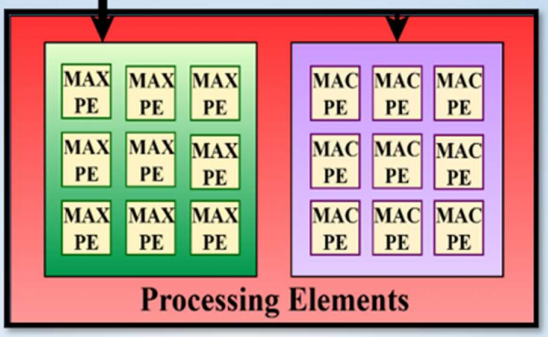
It multiplies an input value with a corresponding weight ie. an image pixel value multiplied by a filter coefficient. The result of the multiplication is then added to an accumulator, which stores the running total of all such operations.



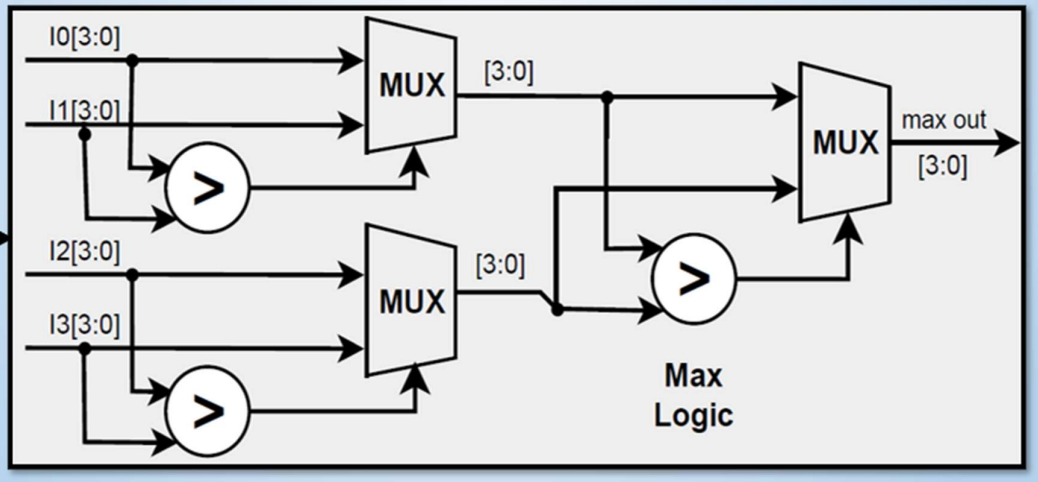
MAC PE



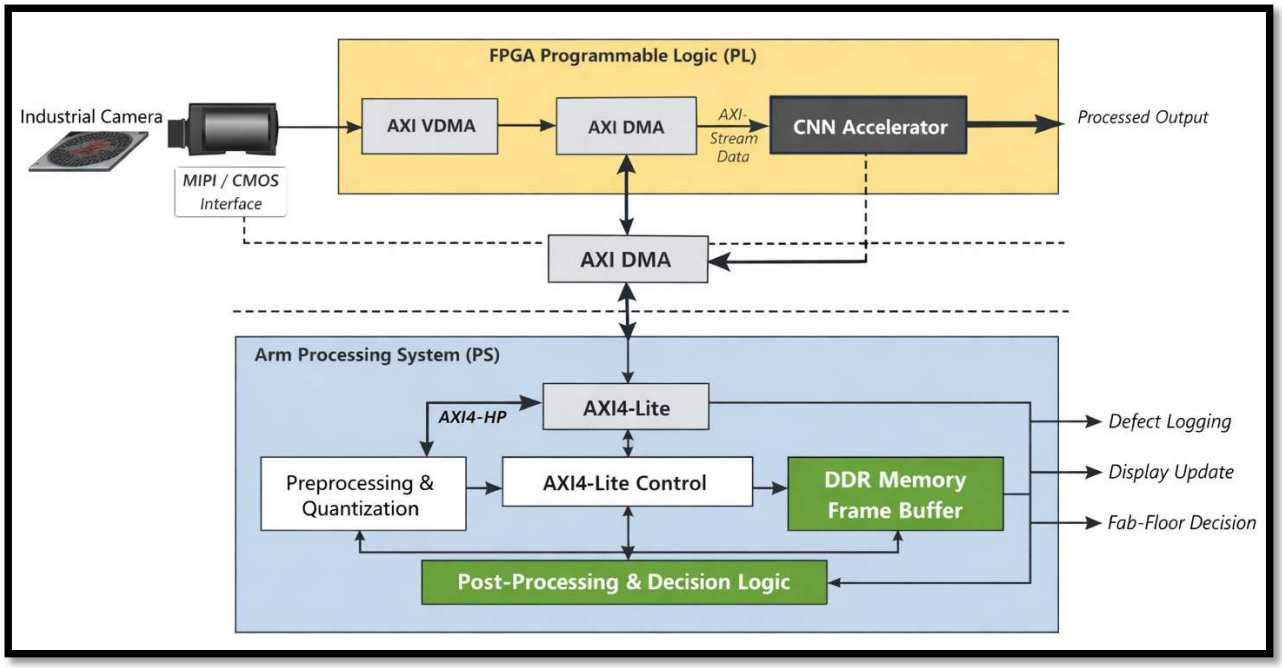
MAX PE



A specialized unit used primarily in pooling operations, where it compares multiple inputs and selects the strongest or most relevant feature in that region.



Integration of CNN Accelerator with PL, PS, and Camera (End-to-End Architecture)
The proposed system is implemented on a Zynq-7000 SoC from Xilinx, which integrates an Arm Cortex-A Processing System (PS) with FPGA Programmable Logic (PL). The system performs real-time wafer defect classification through coordinated interaction between the camera interface, FPGA accelerator, and Arm processor.



1. Camera to Programmable Logic (PL)

Camera Interface

- The wafer image sensor (industrial camera) connects to the PL through:
  - MIPI CSI-2 RX (for high-speed cameras), or
  - Parallel CMOS interface, depending on hardware availability.
- A Camera Receiver IP in the PL converts incoming pixel stream into AXI4-Stream format.

Frame Buffering

- The AXI4-Stream video data is passed to:
  - AXI VDMA (Video DMA)
- VDMA writes image frames into DDR memory through:
  - AXI High-Performance (HP) port of the PS.

  This enables full-frame buffering before inference.

2. Processing System (PS) Role

The Arm Cortex-A processor performs:

- Camera configuration (via I2C/SPI)
- Frame acquisition control
- Image preprocessing:
  - Resizing (if required)
  - Normalization
  - INT8 quantization
- Memory allocation in DDR
- Accelerator configuration via AXI4-Lite
- Post-processing of classification output

The PS accesses DDR and communicates with PL using:

- AXI GP (General Purpose) port → control signals
- AXI HP port → high-speed data transfer

3. PS to CNN Accelerator (Control Path)

The CNN accelerator implemented in PL is exposed as a memory-mapped AXI slave peripheral.
Control Interface (AXI4-Lite)

Used for:

- Start signal
- Reset
- Status monitoring
- Layer parameter configuration
- Interrupt enable/acknowledge

Flow:

1. PS writes input image DDR address to accelerator register.
2. PS writes model parameter base address.
3. PS asserts "start".
4. Accelerator raises interrupt when inference completes.

4. Data Path (High-Speed Path)

1. PS programs AXI DMA with:
    - Source address (input image in DDR)
    - Destination (accelerator input stream)
2. AXI DMA transfers INT8 image to accelerator using AXI4-Stream.
3. Accelerator processes data using:
    - Parallel MAC arrays
    - BRAM-based feature map buffers
4. Output feature/class result is written:
    - Back to DDR via AXI Master interface
    - Streamed to DMA for DDR write-back.

This architecture:

- Minimizes CPU load
- Maximizes throughput
- Enables pipelined execution

5. End-to-End Data Flow

1. Camera captures wafer image.
2. Pixel stream enters PL through MIPI/CMOS interface.
3. AXI VDMA writes frame into DDR.
4. PS preprocesses image and converts to INT8.
5. PS configures accelerator via AXI4-Lite.
6. AXI DMA streams image to accelerator.
7. CNN accelerator performs inference in PL.
8. Output class written back to DDR.
9. PS reads result and performs:
    - Defect logging
    - Display update
    - Fab-floor decision trigger

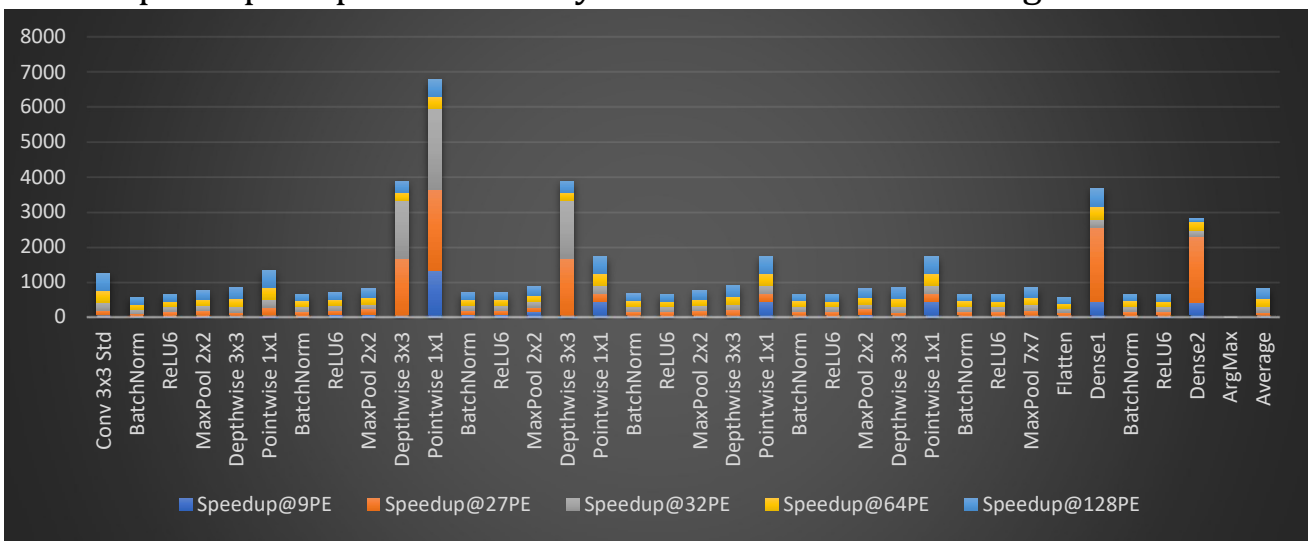
Latency Analysis of Proposed Hardware Accelerator Design with varying (PEs) Processing Element at 100MHz clock (Stored Parameters in On-chip BRAMs)

Layer	Layer Name	Op Type	1PE	9PE	27PE	32PE	64PE	128PE
1	Conv 3x3 Std	MAC	10838016	271642	110412	67139	46674	37025
2	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
3	ReLU6	Elem	25600	6419	2607	1586	1102	874
4	MaxPool 2x2	MAX	48457	1214	493	300	209	165
5	Depthwise 3x3	MAC	3612672	90592	36803	22400	15600	11820
6	Pointwise 1x1	MAC	6422528	16095	65437	39795	27700	21910
7	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
8	ReLU6	Elem	25600	6419	2607	1586	1102	874
9	MaxPool 2x2	MAX	48457	1214	493	300	209	165
10	Depthwise 3x3	MAC	2709504	67795	27606	16787	11657	8849
11	Pointwise 1x1	MAC	7225344	18092	73601	44767	31084	23508
12	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
13	ReLU6	Elem	25600	6419	2607	1586	1102	874
14	MaxPool 2x2	MAX	48457	1214	493	300	209	165
15	Depthwise 3x3	MAC	4064256	10185	41415	25191	17471	13274
16	Pointwise 1x1	MAC	10838016	271642	110412	67139	46674	37025
17	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
18	ReLU6	Elem	25600	6419	2607	1586	1102	874
19	MaxPool 2x2	MAX	48457	1214	493	300	209	165
20	Depthwise 3x3	MAC	338688	8479	3452	2099	1459	1157
21	Pointwise 1x1	MAC	2408448	60243	24537	14936	10377	8226
22	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
23	ReLU6	Elem	25600	6419	2607	1586	1102	874
24	MaxPool 2x2	MAX	48457	1214	493	300	209	165
25	Depthwise 3x3	MAC	677376	16977	6905	4188	2908	2316
26	Pointwise 1x1	MAC	4816896	12068	49063	29860	20725	16454
27	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
28	ReLU6	Elem	25600	6419	2607	1586	1102	874
29	MaxPool 7x7	MAX	121143	3030	1234	750	521	414
30	Flatten	Other	1210	30	12	8	5	4
31	Dense1	MAC	4515840	11314	46075	27898	19460	15425
32	BatchNorm	Elem	51200	12837	5216	3170	2204	1748
33	ReLU6	Elem	25600	6419	2607	1586	1102	874
34	Dense2	MAC	7526400	18860	76717	46612	32290	25705
35	ArgMax	Other	51	1	1	1	1	1
36	Total		11,48,90,005	28,79,500	11,70,405	7,12,039	4,94,909	3,92,314

Speedup Analysis of Proposed Hardware Accelerator Design with varying (PEs) Processing Element at 100MHz clock with 1PE design

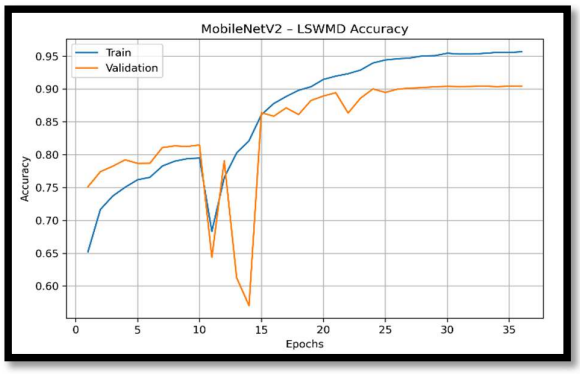
Layer	Layer Name	Op Type	Speedup@9PE	Speedup@27PE	Speedup@32PE	Speedup@64PE
1	Conv 3x3 Std	MAC	54.75×	139.05×	228.03×	329.28×
2	BatchNorm	Elem	30.41×	69.92×	116.36×	153.90×
3	ReLU6	Elem	28.07×	116.36×	154.54×	153.35×
4	MaxPool 2x2	MAX	31.99×	142.99×	142.93×	196.18×
5	Depthwise 3x3	MAC	36.01×	97.78×	165.02×	223.99×
6	Pointwise 1x1	MAC	43.46×	230.84×	230.84×	349.62×
7	BatchNorm	Elem	30.30×	116.96×	154.22×	153.91×
8	ReLU6	Elem	70.33×	116.36×	153.29×	153.35×
9	MaxPool 2x2	MAX	83.84×	142.51×	142.79×	196.18×
10	Depthwise 3x3	MAC	36.00×	1646.63×	1650.00×	223.80×
11	Pointwise 1x1	MAC	1332.05×	2308.00×	2307.90×	349.62×
12	BatchNorm	Elem	69.89×	116.36×	154.22×	153.91×
13	ReLU6	Elem	69.73×	116.36×	154.55×	153.35×
14	MaxPool 2x2	MAX	142.87×	142.51×	142.79×	196.18×
15	Depthwise 3x3	MAC	36.01×	1650.00×	1650.00×	223.80×
16	Pointwise 1x1	MAC	433.96×	230.41×	230.42×	349.89×
17	BatchNorm	Elem	30.26×	118.00×	154.54×	155.26×
18	ReLU6	Elem	28.10×	115.84×	153.29×	153.25×
19	MaxPool 2x2	MAX	31.88×	142.79×	142.79×	196.18×
20	Depthwise 3x3	MAC	36.00×	164.75×	165.00×	223.99×
21	Pointwise 1x1	MAC	434.31×	230.81×	231.35×	349.62×
22	BatchNorm	Elem	30.30×	116.36×	154.22×	153.91×
23	ReLU6	Elem	28.07×	116.36×	154.55×	153.35×
24	MaxPool 2x2	MAX	83.84×	142.51×	142.79×	196.18×
25	Depthwise 3x3	MAC	36.00×	103.69×	164.95×	223.89×
26	Pointwise 1x1	MAC	434.50×	231.15×	231.21×	349.62×
27	BatchNorm	Elem	30.30×	116.36×	154.22×	153.91×
28	ReLU6	Elem	28.07×	116.36×	154.55×	153.35×
29	MaxPool 7x7	MAX	36.02×	153.71×	154.15×	210.02×
30	Flatten	Other	28.14×	110.00×	110.00×	134.44×
31	Dense1	MAC	439.51×	2130.00×	228.11×	350.33×
32	BatchNorm	Elem	30.30×	116.36×	154.22×	153.91×
33	ReLU6	Elem	28.07×	116.36×	154.55×	153.35×
34	Dense2	MAC	400.00×	1890.00×	186.35×	255.00×
35	ArgMax	Other	1.19×	4.13×	4.64×	5.67×
36	Average	-	39.88×	98.13×	161.43×	232.12×

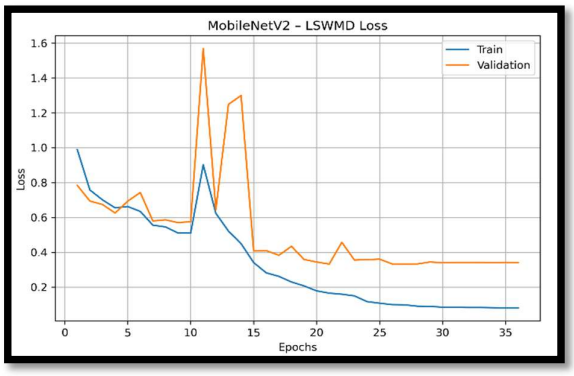
Speedup Graph for each Layer with different PE configuration

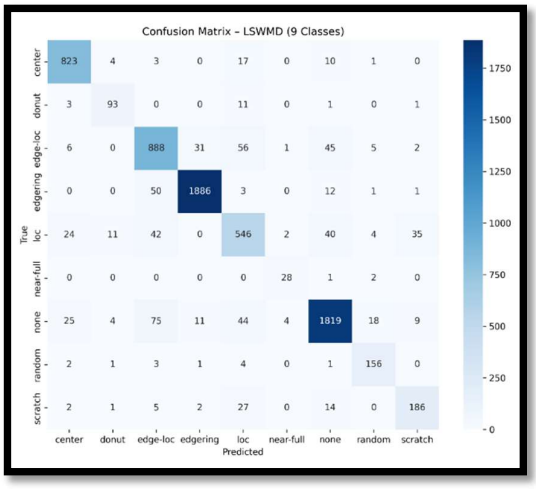


PEs	Total Latency (cycles)	Speedup	Latency (ms)
1	11,48,90,005	1×	1148.9
9	28,79,500	39.88×	28.8
27	11,70,405	98.13×	11.7
32	7,12,039	161.43×	7.12
64	4,94,909	232.12×	4.95
128	3,92,314	292.79×	3.92

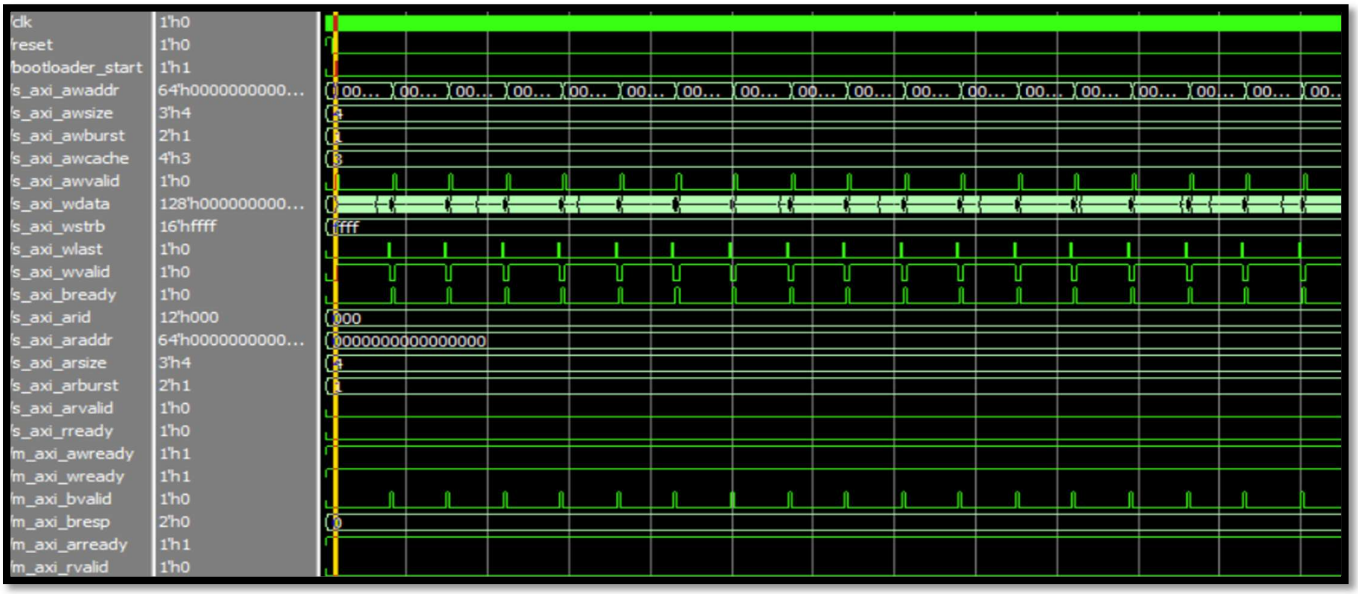
Training Accuracy and Loss of the Dataset using MobilenetV2



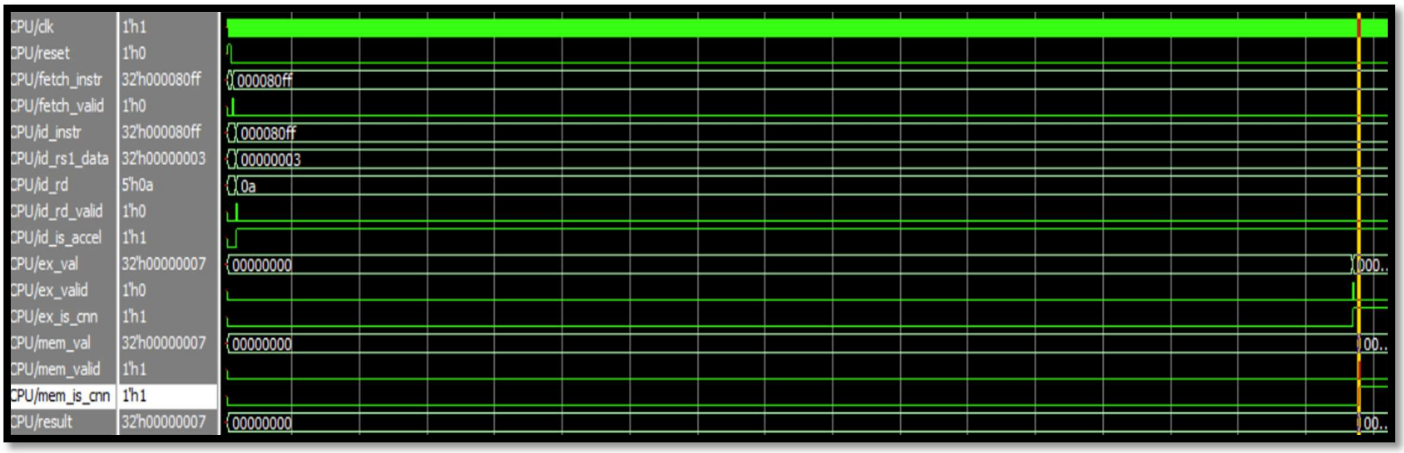




Simulation 1 : Loading of Image Input from BootRom to SoC and sending it to Accelerator Memory Ram via AXI Interface in Burst Mode (Stored Parameters in On-chip BRAMs)

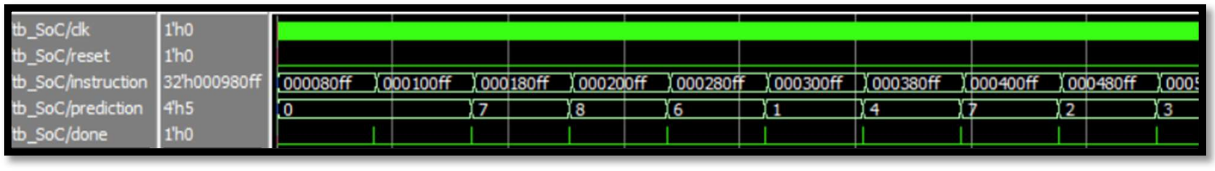


Simulation 2 : Handshaking from Processor module to Accelerator



Simulation 3 : Reception of 'Done' signal from the Accelerator Engine along with the predicted defect classification for the silicon wafer image

Simulation 5 : Complete Post Implementation Simulation of Classification of 20 Silicon Wafer Defect Images



Hardware Implementation with Output Details

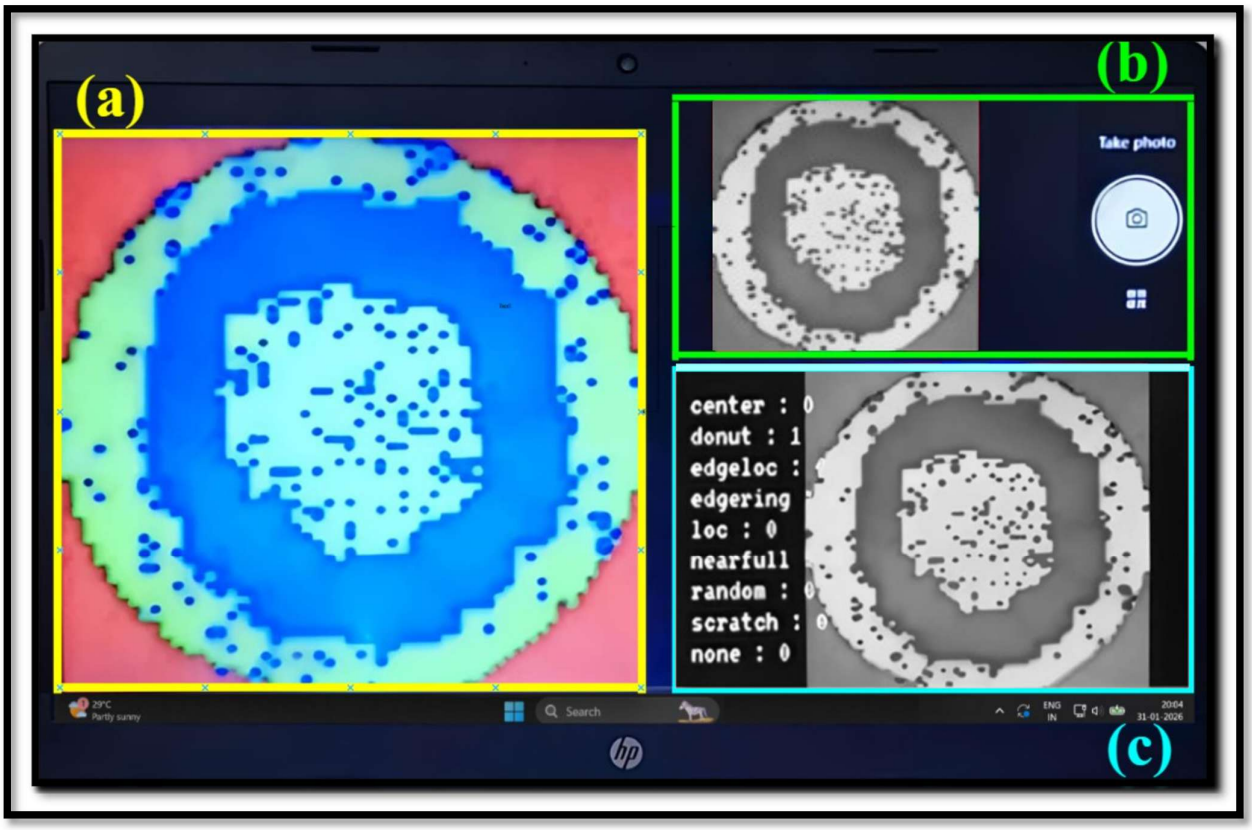
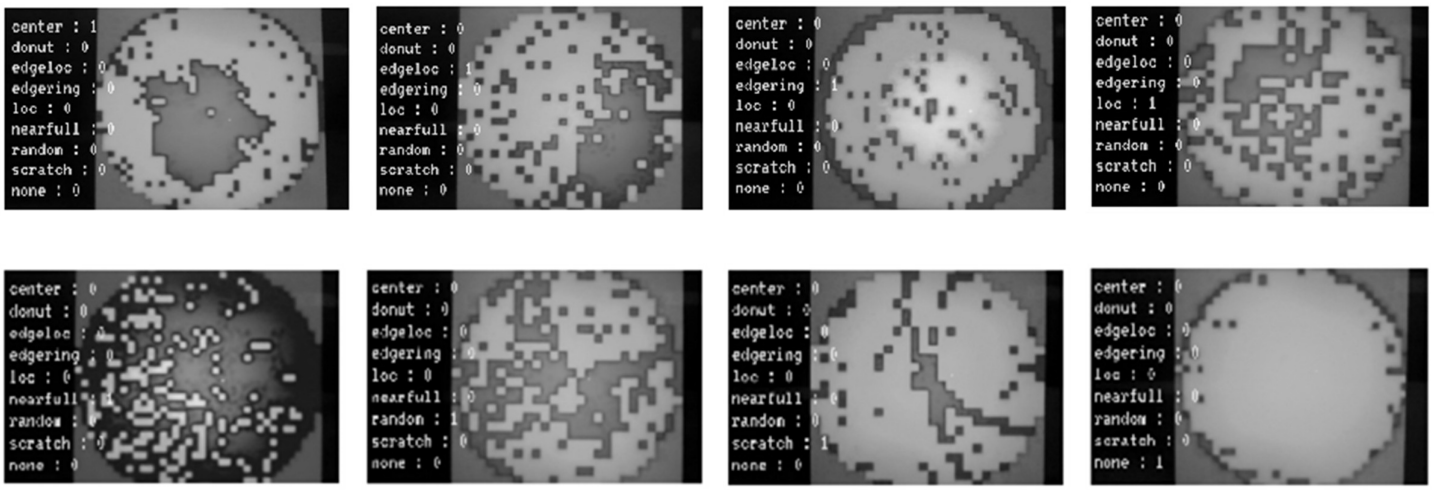


Fig. a : The left-side image serves as the input image displayed to camera, which captures the visual data for processing. The processing board and camera module are positioned to enable real-time image acquisition for ML inference.

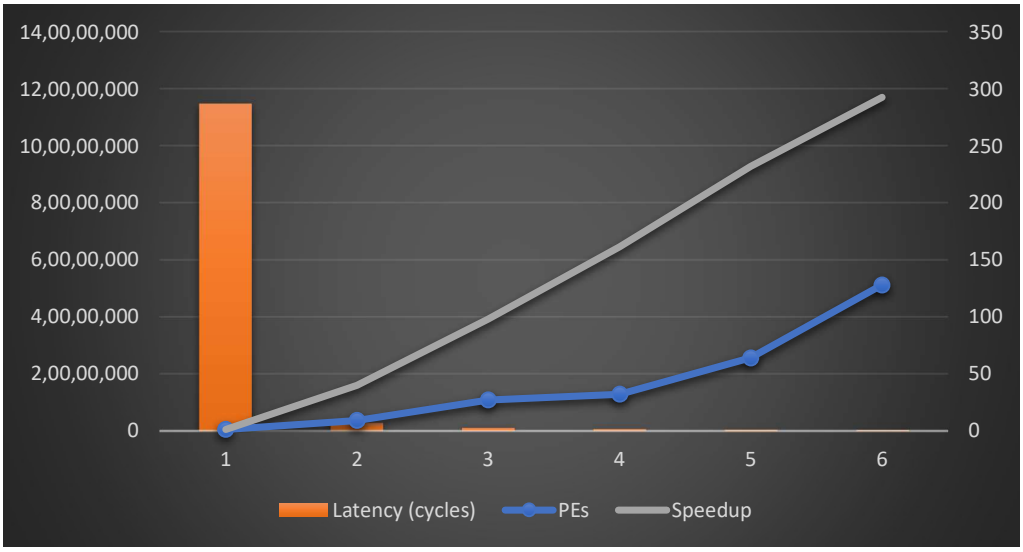
Fig. b : The right-top panel shows the image captured by the camera.

Fig. c : The right-bottom panel illustrates the output prediction produced by the ML accelerator.

All Silicon Wafer Defect Image Detection Inference



Results



Speedup vs Latency wrt. PEs

Hardware Resources Utilization

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	39947	53200	75.08
FF	60532	106400	56.89
BRAM	117	140	83.57
DSP	72	220	32.72
IO	87	200	43.50
BUFG	8	32	25.00

Power Analysis

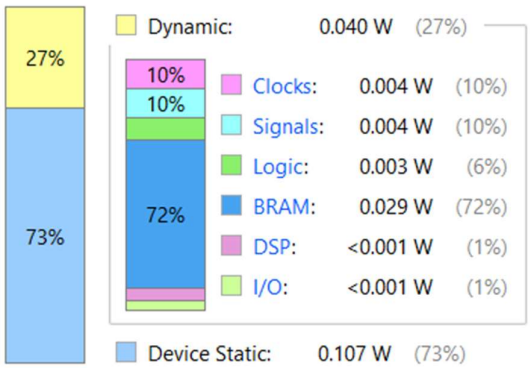
Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.147 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.7°C**
Thermal Margin:	58.3°C (4.9 W)
Effective θJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.110 ns	Worst Hold Slack (WHS):	0.049 ns	Worst Pulse Width Slack (WPWS):	11.250 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	9952	Total Number of Endpoints:	9952	Total Number of Endpoints:	2802

All user specified timing constraints are met.

Final Results

SR. NO.	METRIC	VALUE
1	THROUGHPUT	255 FPS
2	ACCURACY	95.52% (INT8)
3	LATENCY PER FRAME (CPU)	1148.9 ms/frame
4	LATENCY PER FRAME (FPGA)	3.92 ms/frame
5	SPEED UP (w.r.t. cycles)	293.09
6	LOOK-UP TABLES (LUTs)	39947 (75.08%)
7	FLIP FLOP (FF)	60532 (56.89%)
8	BRAM : DSP	75 : 87
9	ON CHIP POWER	0.147 Watt
10	ENERGY PER FRAME	368 µJ/frame

Conclusion

✓ Why Deep Learning for Wafer Defect Detection?

Deep learning, particularly Convolutional Neural Networks (CNNs), enables automatic feature extraction from wafer images without relying on handcrafted rules. It effectively handles complex and diverse defect patterns present in industrial datasets such as WM811K, maintaining robust classification performance even under high variability. Compared to traditional machine learning approaches, CNN-based methods improve generalization, scalability, and adaptability for real-time semiconductor inspection systems.

✓ Why MobileNetV2 Specifically?

MobileNetV2 is selected due to its lightweight architecture and computational efficiency. Its use of depthwise separable convolutions and inverted residual blocks significantly reduces multiply–accumulate (MAC) operations while preserving high classification accuracy. The architecture is highly suitable for FPGA acceleration because convolution operations can be parallelized efficiently in programmable logic. Its compact design makes it ideal for deployment on resource-constrained edge platforms such as the Zynq SoC, where power, memory, and logic utilization must be carefully managed.

✓ Model Design for WM811K Dataset

The model processes high-resolution wafer images obtained from inspection equipment. Using Quantization-Aware Training (QAT), the FP32 baseline achieved 97% classification accuracy. After INT8 quantization, the model achieved 95.5% accuracy while reducing memory footprint and computational cost by approximately 75%. The quantized model is deployed on the FPGA accelerator, and predicted defect classes are communicated to the Arm Processing System (PS) for further decision-making and logging.

✓ Custom AI Accelerator with Pipeline-Mapped MobileNetV2

The FPGA accelerator, implemented in the Programmable Logic (PL) of the Zynq SoC, maps standard convolution, depthwise convolution, pointwise convolution, batch normalization, max pooling, and fully connected layers onto parallel MAC/MAX-based processing elements. Both intra-layer and inter-layer pipelining techniques are employed to minimize latency and maximize hardware utilization. The accelerator is integrated with the Arm Cortex-A Processing System through AXI4-Lite (control) and AXI DMA (high-speed data transfer), ensuring efficient communication and synchronized execution within the SoC.

✓ Performance and Deployment

The complete inference pipeline—including feature extraction, dense computation, argmax, and classification—executes in approximately 3,92,314 cycles at 100 MHz, corresponding to 3.92 ms latency per frame. The accelerator achieves approximately 255 FPS throughput at 0.147 Watt Power on a ~40% utilization of board resources, meeting real-time inline inspection requirements. A significant speedup over software-only INT8 inference is achieved due to efficient depthwise–pointwise convolution mapping and pipelined execution in hardware. Computation is performed primarily using on-chip BRAM buffers to reduce external memory dependency and avoid bandwidth bottlenecks. The Arm processor manages accelerator control, result retrieval, logging, display updates, and fab-floor decision triggering without reliance on external PCs, GPUs, or cloud services, ensuring deterministic execution and data security. The modular accelerator design supports scalability to higher-resolution wafer images or expanded channel depths through parallel tiling strategies while maintaining architectural flexibility.

References

1. *A. Su Pan, B. Xingyang Nie, C. Xiaoyu Zhai; Enhancing wafer defect detection via ensemble learning. AIP Advances 1 August 2024; 14 (8): 085301.* https://doi.org/10.1063/5.0222140
2. *Mayank Jariya, Parveen Kumar, Rekha Devi, Balwinder Singh, Silicon wafer defect pattern detection using machine learning,Materials Today: Proceedings,2023,ISSN 2214-7853,* https://doi.org/10.1016/j.matpr.2023.04.233.3
3. *R. K.P. and S. V., "Machine Learning Approach for Mixed type Wafer Defect Pattern Recognition by ResNet Architecture," 2023 International Conference on Control, Communication and Computing (ICCC), Thiruvananthapuram, India, 2023, pp. 1-6, doi: 10.1109/ICCC57789.2023.10165078.*
4. *Wu, Di & Zhang, Yu & Jia, Xijie & Tian, Lu & Li, Tianping & Sui, Lingzhi & Xie, Dongliang & Shan, Yi. (2019). A High-Performance CNN Processor Based on FPGA for MobileNets. 136-143. 10.1109/FPL.2019.00030.*
5. *W. Jiang, H. Yu and Y. Ha, "A High-Throughput Full-Dataflow MobileNetv2 Accelerator on Edge FPGA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 5, pp. 1532-1545, May 2023, doi: 10.1109/TCAD.2022.3198246.*
6. *Huang, J.; Liu, X.; Guo, T.; Zhao, Z. A High-Performance FPGA-Based Depthwise Separable Convolution Accelerator. Electronics* ***2023****, 12, 1571.* https://doi.org/10.3390/electronics12071571
7. *Sandler, Mark & Howard, Andrew & Zhu, Menglong & Zhmoginov, Andrey & Chen, Liang-Chieh. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 4510-4520. 10.1109/CVPR.2018.00474.*
8. *A. Su Pan, B. Xingyang Nie, C. Xiaoyu Zhai, Title: Enhancing wafer defect detection via ensemble learning, Journal: AIP Advances, August 2024*
9. *Mayank Jariya, Parveen Kumar, Rekha Devi, Balwinder Singh, Title: Silicon wafer defect pattern detection using machine learning, Journal: Materials Today: Proceedings, 2023*
10. *R. K.P. and S. V., Title: Machine Learning Approach for Mixed type Wafer Defect Pattern Recognition by ResNet Architecture, Conference: International Conference on Control, Communication and Computing (ICCC), 2023*
11. *Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, Title: MobileNetV2: Inverted Residuals and Linear Bottlenecks, Conference: CVPR 2018,*
12. *Guangyuan Deng and Hongcheng Wang Efficient Mixed-Type Wafer Defect Pattern Recognition Based on Light-Weight Neural Network by Guangyuan Deng and Hongcheng Wang School of Electrical Engineering and Intelligentization, Dongguan University of Technology, Dongguan 523808, China*
13. *M. Saqlain, Q. Abbas and J. Y. Lee, "A Deep Convolutional Neural Network for Wafer Defect Identification on an Imbalanced Dataset in Semiconductor Manufacturing Processes," in IEEE Transactions on Semiconductor Manufacturing, vol. 33, no. 3, pp. 436-444, Aug. 2020, doi: 10.1109/TSM.2020.299435*
14. *Efficient FPGA-Based Convolutional Neural Network Implementation for Edge Computing Cuong Pham-Quoc 1,2,* and Tran Ngoc Thinh 1,2 1Department of Computer Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam*
15. *A Scalable Architecture for CNN Accelerators Leveraging High-Performance Memories Maarten Hattink , Giuseppe Di Guglielmo , Luca P. Carloni , and Keren Bergman*