Wafer Automatic Defect Classification *Using FPGA accelerator*

Dongbin Shin, SeungJu Byun, KangHyeon Cha and EunSeong Lee
Department of Electronic Engineering, Kwangwoon University, South Korea
E-mail: dongbin4013@kw.ac.kr
Advisor: Prof. Seoung-Jun Oh

***Abstract*— Wafer bin map은 많은 데이터를 가지고 있다. 이를 효율적으로 처리하기 위해 딥러닝 기술을 도입하고 있다. VGG-16과 거리측정 방식을 GPU상에서 작동시키는 경우가 많다. 하지만 CNN은 계산능력때문에 높은 전력 소비량을 요구한다. FPGA는 GPU 대비 낮은 전력을 소모한다. 따라서 본 연구에서는 VGG-16과 거리측정 방식을 FPGA로 가속연산을 하여 높은 전력효율을 얻고자 한다.**

***Index Terms*—Wafer Bin Map, FPGA, CNN, VGG-16, Kibana**

I. Introduction

반도체 생산 과정에서 wafer 한 장에는 많은 시간과 자원이 투자된다. 그렇기 때문에 반도체 수율은 기업의 경쟁력 핵심 요소다. 불량에 대한 빠른 원인 파악 및 개선작업을 통해 고수율을 유지하는 것이 매우 중요하다.[1][2] 반도체 수율을 높이기 위해서 불량패턴을 분석한다. 특정패턴을 형성하는 불량은 특정 공정의 문제를 나타낼 가능성이 많고 또한 같은 불량패턴의 반도체 칩 역시 동일한 원인을 가지고 있을 확률이 높다. 현재 대부분의 기업에서 Wafer Bin Map(WBM)불량 패턴분류를 엔지니어들이 사후분석을 하기 때문에 엔지니어간 역량에 의해 차이가 발생한다. 또한 높은 반도체 수요에 맞추어 많은 엔지니어를 투입하는 데는 비용과 시간문제가 있다.[3]

이를 극복하기위해 딥러닝을 활용한 연구가 활발히 진행중이다. Park et al. (2018)[4] 은 CNN을 이용해 WBM분류를 제안했다. 하지만 이 연구에서는 WBM의 신규패턴분류를 할 수 없다는 단점이 있다. 새로운 패턴을 분류 추출해내지 못하면 다양한 원인을 분석, 해결하지 못하게 된다. Cheon et al. (2019)[5] 는 CNN과 거리측정 알고리즘을 결합하여 WBM을 분류하는 방법을 제안했다. CNN을 통해 기존 불량패턴을 분류하고 미분류된 불량 패턴에 대해 거리 측정 방식인 유클리디안 거리 측정 방식을 활용한 군집화를 제안했다. 이러한 방식은 안정적인 ADC의 성능을 보장한다.

기존 GPU 딥러닝시 컨볼루션 레이어에서 전파 처리시간의 대부분이 소요된다. 그에 따라 전력소비량이 늘어난다. FPGA는 데이터 병렬화 및 누적 곱셈 연산에서 GPU 대비 우수한 성능과 전력소모를 보인다. 따라서 2D 콘볼루션 가속화 레이어 부분을 FPGA 가속기를 설계하여 해결한다면 연산시간과 전력소모에서 이득을 얻을 수 있다.[6] 따라서 본 연구에서는 Wafer Automatic Defect Classification을 사전 학습한 뒤 weight를 추출하여 FPGA 가속기를 설계할 것이다. 이후 미분류 WBM 패턴에 대해 거리측정 방식으로 후처리를 진행할 것이다.

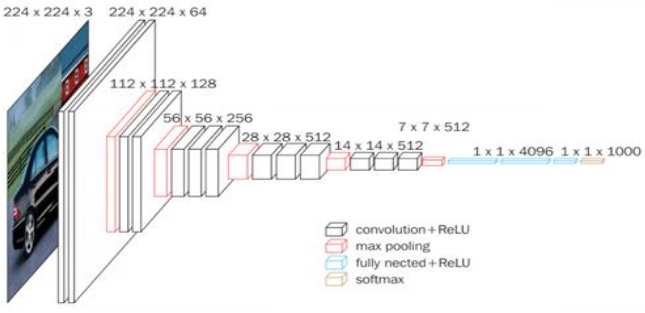
II. Proposed Design

A. Dataset

본 연구에서 사용한 데이터는 Binary map으로 오픈되어 있고 가장 많이 사용하는 WM-811K 데이터셋을 사용한다. 811,457장의 웨이퍼맵으로 46,393로트의 실제 Fabrication에서 수집한 것이다. 각 이미지는 26 * 26으로 구성되어 있다.

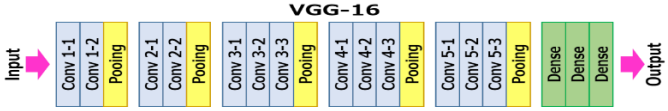
B. VGG-16(Visual Geometry Group from Oxford)

VGG-16 Overview



VGG-16은 16개 계층으로 구성된 컨벌루션 신경망이다. 비선형성을 증가시키기 위해 매개 변수의 수를 줄여 모든 합성곱 레이어에서 3*3 필터를 사용해 layer를 여러 개 쌓을 수 있다. 일반적으로 layer의 깊이가 깊어질수록 물체의 특징을 세부적으로 검출하기에 16개 계층으로 이뤄진다.

[VGG-16 Flow]



VGG-16 구조의 구성은 13 convolution layers(특징 추출)+3 fully-connected layers(분류 과정), 3*3 convolution filters, stride: 1& padding: 1, 2*2 max pooling(stride:2), ReLU로 구성한다.

WBM의 경우 VGG-16 구조가 Feature map 검출에 가장 특화되어 있음을 선행 연구에서 확인하였다. (S.cheon 2019)

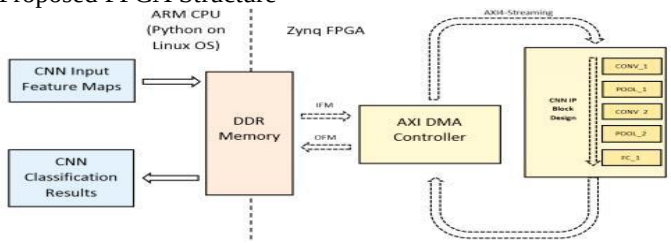
C. Post processing

VGG-16을 통한 분류 이후에 아직 분류가 안 된 wafer들에 대해 한 번 더 분류를 하는 작업을 위한 것이다. 따라서 거리를 측정하는 방식을 설정한 거리(임계점) 이상에 존재하는 것들에 대해서 새로운 class를 정해주는 방식을 통해 한 번 더 미분류 된 것들에 분류하는 과정을 거친다. 거리는 점과 점 사이 최단 거리를 구하는 방법인 유클리디안 거리 측정 방식을 사용했다. 유클리디안 거리 측정 식은 아래의 식과 같다.

$$\sqrt{(Ax - Bx)^2 + (Ay - By)^2}$$

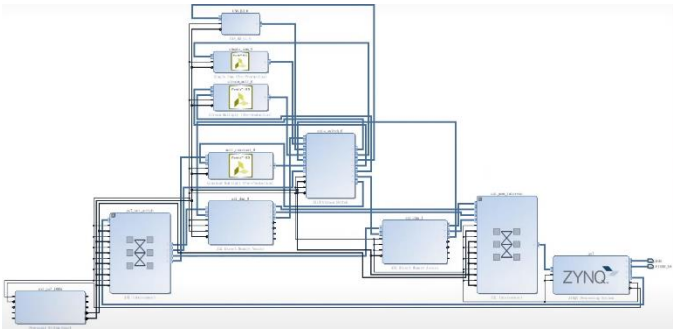
D. FPGA

Proposed FPGA Structure

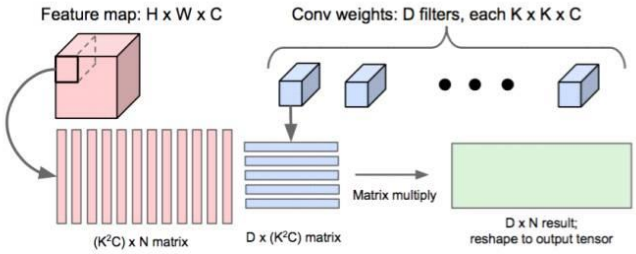


Host PC 에서의 ARM 리눅스 와 Zynq FPGA 구현 두 부분으로 나뉜다. Linux에서 python과 딥러닝 프레임워크로 제어되는 부분을 high level로 두고, 프레임워크가 보드의 DDR 메모리에서 입력 feature 들을 load 하고 다시 classification 결과를 출력한다. DMA controller API를 이용한 뒤, Verilog로 작성한 CNN IP를 디자인하여 high speed forward propagation을 수행한다. 이는 고성능의 CNN 구축을 목표로 한 가속기 구조이다[7]

[IP Design Diagram] (in Vivado 2020.02)



1. im2col[8]



2. Pooling Layer

3. Fully-connected Layer

ID	Batch	Convolution (커널)	input feature map (채널)	input feature map dimension	output feature map (채널)	output feature map dimension	padding dimension

4. data(parameter) flow

[사용할 API 드라이버]

프레임워크 아래 DDR 에서의 데이터 전송 요청으로 DDR에서 FPGA 내장 BRAM으로 이동하게 한다.

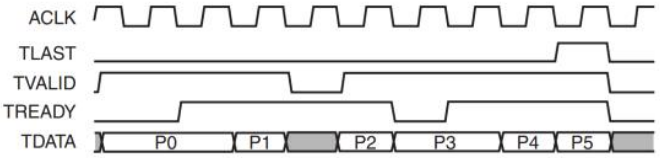
1. MMIO (memory-mapped input/output)

메모리 엑세스 프로토콜로, 포인터는 FPGA의 컨트롤/상태 레지스터의 포인터를 initialize 해준 뒤, MMIO는 FPGA에 포인터를 통해 로드, 저장한다.

2. DMA (Direct Memory Access)

직접 메모리 엑세스로, 데이터의 전송은 FPGA의 내장 컨트롤러를 통해 독립적으로 제어, 예약된다. 이는 최대 처리량으로 전송된다. 제공되는 AXI DMA 는 AXI4-Streaming 프로토콜에서 데이터 스트림을 전송한다.

[AXI4-streaming 인터페이스의 waveform]



클럭 사이클 당 한 Word의 스트리밍 속도를 지원(1개의 레이어 IP에서 1 클럭 사이클 당 1개의 word를 사용하고, 새로운 word 마다 출력한다.)[9]

[Framework]

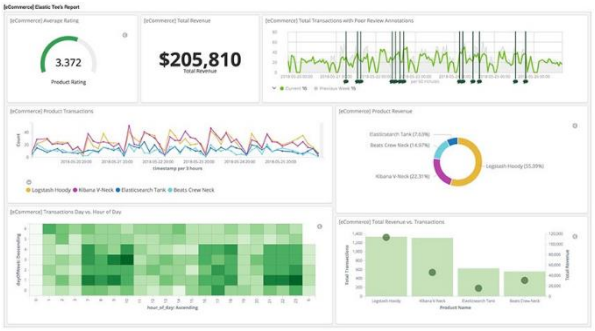
Caffe, TensorFlow, Theano 중, PYNQ linux OS에서 빌드했을 때, cross-platform에서 가장 호환성이 좋다고 나타나는 Theano, caffe 프레임워크를 사용할 예정이다. (Wei Dai. 2019)

[FPGA-CNN Accelerator 실행 예제][10]

```
net = {}
# Input image with dimension 28 x 28
net['input'] = InputLayer((None, 1, 28, 28))
net['lenet'] = FPGA_LENET(FPGA_net['input'])
# FPGA_LENET 은 커스톰 된 레이어이며, 제공하는 API와
# FPGA IP 의 실행을 호출한다.
```

E. Visualize(Kibana)

Kibana란 Elastic Stack 기반으로 구축된 오픈 소스 프론트엔드 애플리케이션이다. ElasicSearch에서 색인된 데이터를 검색하여 시각화 및 분석하는 기능을 한다. 히스토그램, GEO맵, 차트 등 다양한 시각화 도구를 이용하여 사용자가 지정한 대시보드와 결합해 데이터 가독성을 높여준다.



(https://www.elastic.co/kr/kibana/features)

위 그림과 같이 다양한 시각화 도구를 통해 빅데이터를 가독성 높은 서비스 형태로 사용자에게 제공할 수 있다. 본 연구에서는 분석을 마친 wafer data들에 대해 그 결과를 단순한 콘솔창이 아닌 Kibana형식으로 제공할 것이다. 이로 인해 classification을 마친 데이터를 확인하는 엔지니어 간의 해석차이를 줄이고 편의성을 증가시킬 수 있다.

III. 성능 평가

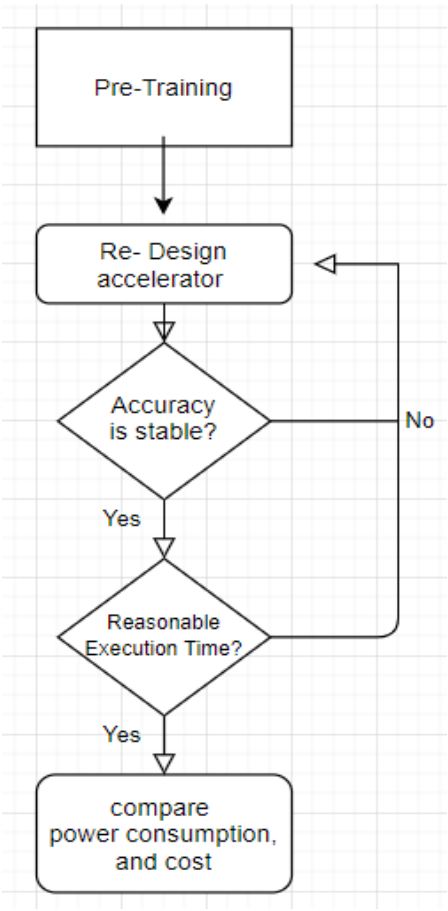
System	i5-10400F GTX 1060 3gb	i5-10400F RTX 2070	pynq z2
cost (msrp)	$157 / $249	$157 / $599	$119

이번 연구에서 사용되는 FPGA는 PYNQ-Z2로 MSRP는 199$이다. 또한 GPU의 경우 GTX1060-3GB과 RTX-2070는 각각 $199와 $599이다. 이와 같이 가격이 3배 차이나는 GPU를 넣은 이유는 FPGA와 가격이 비슷한 GPU와의 비교를 통해 가격과 전력 대비 효율 차이가 없다는 것을 확인하고자 사용했다.

FPGA accelerator가 기존 GPU accelerator에 대비해 비슷한 run time 및 accuracy를 도출해낼 때 서로 얼만큼의 가격, 평균 소비 전력, 최대 소비 전력이 차이가 나는 지 측정할 것이다.

소비자 가격의 경우 공급과 수요에 따라 변동성이 있으므로 MSRP를 기준 가격으로 설정한다. MSRP는 제조사의 권장 소비자 가격을 말하는 데 이 가격은 변동성 없는 표준 가격이므로 이를 기준으로 한다. 전력 측정의 경우 순간 최대 소비 전력(peek)과 평균 전력을 측정하여 비교할 것이다.

<Project Flow Chart>



IV. Conclusion

본 연구에서는 가속기로 GPU 대신 FPGA를 사용했을 때 정확도와 시간측면에서 비슷하도록 유지하면서, 동시에 전력 대비 효율 및 비용에서 우세함을 보일 것으로 예상된다. FPGA와 GPU를 최대한 비슷한 가격의 FPGA와 GPU와 가격이 3배 정도 차이나는 GPU인 RTX-2070와의 비교를 통해 FPGA가 전력 대비 효율과 가격 측면에서 우세할 것이다. 따라서 결과적으로 얻게 되는 이점은 전력효율 측면에선 FPGA가 GPU에 비해 50배 이상으로 차이가 나고 GPU(RTX-2070) 인 경우 가격 측면에서도 2배 이상의 차이를 보일 것이다.

V. References

[1] Han Young Shin, Hwang Mi Young, Lee Chil Gee, (2003) Automatic classification of failure patterns in semiconductor EDS Test using pattern recognition, The Institute of Electronics and Information Engineers 2003.7,703-706

[2] Lee Chang Hyun, Ki Seoung Bum, (2019) Identifying Wafer Defect Patterns by Variational Autoencoder and SegNet, Journal of the Korean Institute of Industrial Engineers 45(2), 117-124

[3] Liu, C.-W. and Chien, C.-F. (2013), An intelligent system for wafer bin map defect diagnosis: An empirical study for semiconductor manufacturing, Engineering Applications of Artificial Intelligence, 26(5),1479-1486.

[4] Jaesun Park, Junhong Kim, Hyungseok Kim, Kyounghyun Mo, Pilsung Kang, (2018) Wafer Map-based Defect Detection Using Convolutional Neural Networks, Journal of the Korean Institute of Industrial Engineers 44(4), 2018.8, 249-258(10 pages)

[5] Sejune Cheon, Hankang Lee, Chang Ouk Kim, Seok Hyung Lee, (2019) Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class, IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING, VOL. 32, NO. 2, MAY 2019

[6] Chao Huang, Siyu Ni, Gengsheng Chen, (2017) A Layer-based Structured Design of CNN on FPGA, 2017 IEEE 12th International Conference on ASIC (ASICON),10.1109/ASICON.2017.8252656

[7] Xilinx. Vivado high-level synthesis.

[8] Ben Cope Implementation of 2d convolution on fpga, gpu and cpu.

[9] Xilinx. Axi reference guide. 2011.

[10]Hongxiang Fan, Martin Ferianc, Shuanglong Liu, Zhiqiang Que, Xinyu Niu, Wayne Luk, (2020) Optimizing FPGA-Based CNN Accelerator Using Differentiable Neural Architecture Search, 2020 IEEE 38th International Conference on Computer Design (ICCD), 10.1109/ICCD50377.2020.00085