ECE284 Fall 21 W1S1

Low-power VLSI Implementation for Machine Learning

Prof. Mingu Kang

Introduction – Affiliation / Education

Mingu Kang

- **Assistant Professor** **2020**
  - ECE department, UC San Diego (UCSD)
- **Ph.D. UIUC, ECE, IL, USA** **2017**
  - Academic advisor: Naresh Shanbhag
  - Thesis: Deep In-memory Computing
- **B.S. / M.S. Yonsei Univ., EE, Korea** **2007 / 2009**
  - Academic advisor: Seong-Ook Jung
  - Thesis: Low-power SRAM and Content Addressable Memory

Professional Career

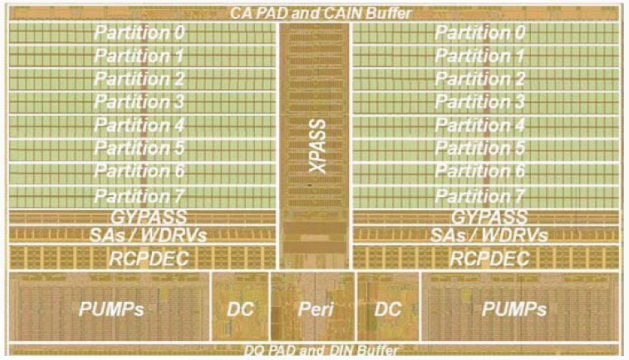
- Prior experience:

IBM TJ Watson Research Center, NY, 2017 – 2020

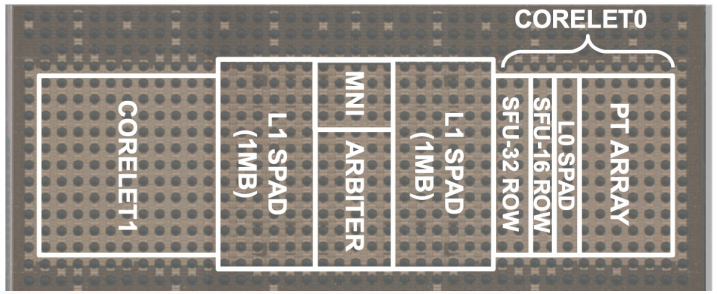
- Machine learning accelerator architecture team
- Research staff member

Samsung Semiconductor, Korea, 2009 – 2012

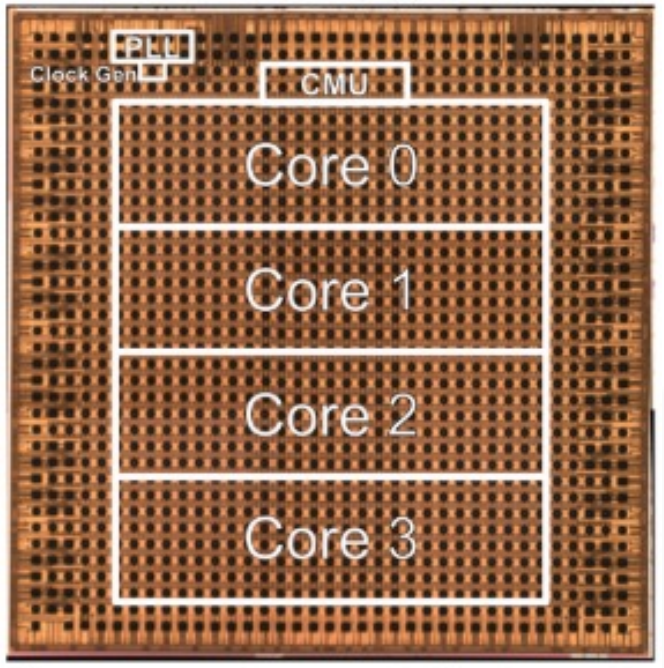
- Phase change memory (PCM / PRAM) team
- Research & development engineer



8Gb PRAM in 20nm (Samsung)

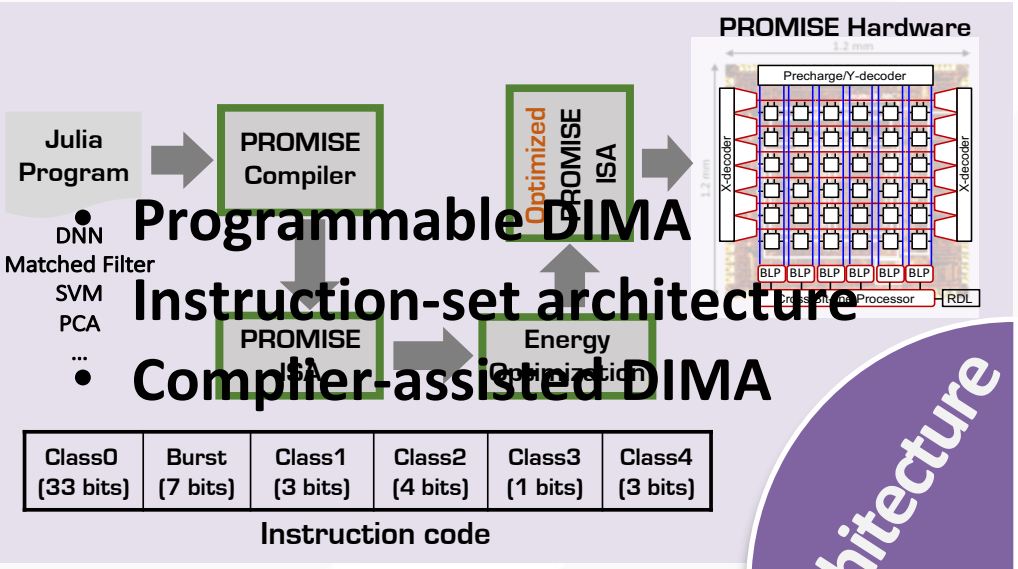


Scalable AI Processor in 14nm (IBM)
VLSI20

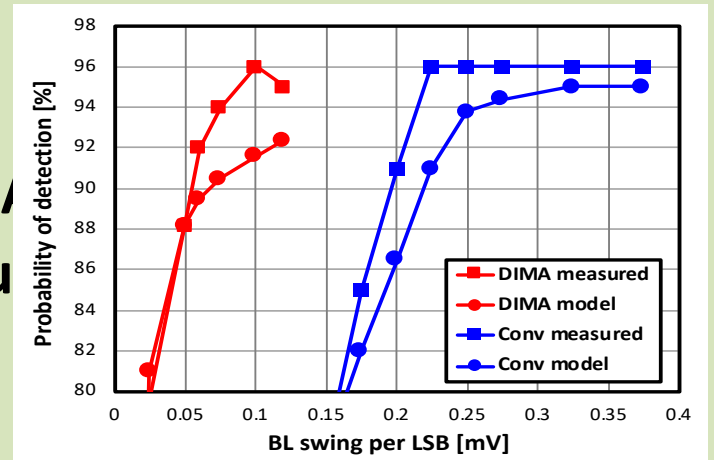


Scalable AI Processor in 7nm (IBM)
ISSCC21

Vertically-Integrated Research in Deep In-memory Computing

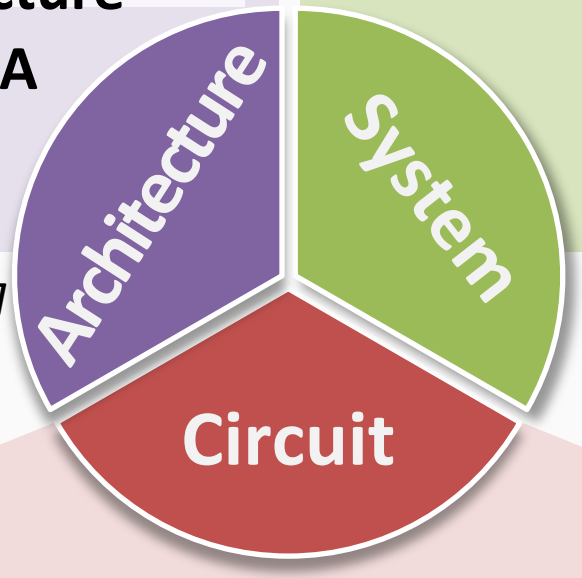


- **Programmable DIMA**
- **Instruction-set architecture**
- **Compiler-assisted DIMA**



- DIMA
- Accu

$$p_{\text{miss,DIMA}} = Q\left(N\alpha_{\text{SVM}}\sqrt{\text{SNR}_{\text{DIMA}}}\right)$$

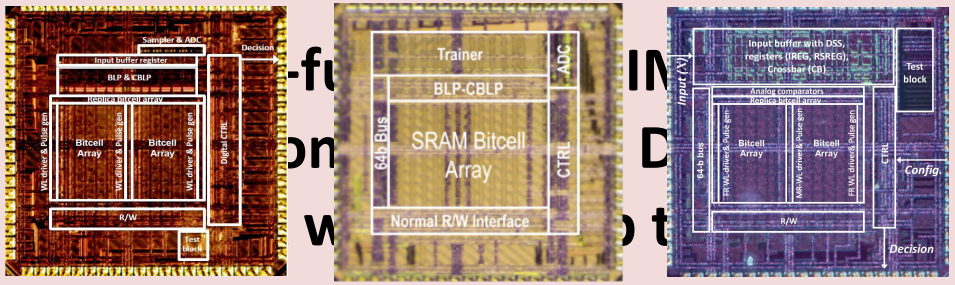


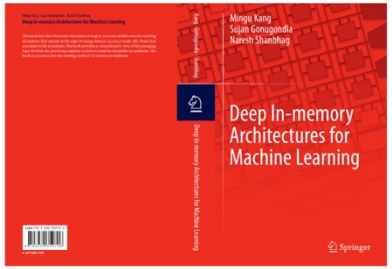
- M.Kang*, P.Srivastava*, et al., ***[IEEE MICRO 19]***
- P.Srivastava*, M.Kang*, et al., ***[ISCA 18]***
  [IEEE MICRO TOP Pick 19 Honorable Mention]
  (*equal contribution)

- M.Kang et al.,
  [TCAS1 20] [ICASSP 14,15] [TBIOCAS 16]
- Y.Kim, M.Kang et al.,***[TCOMM 17] [ISIT 17]***

- M.Kang et al.,
  [JSSC 18Jan] [JSSC 18Jul] [ESSCIRC 17]

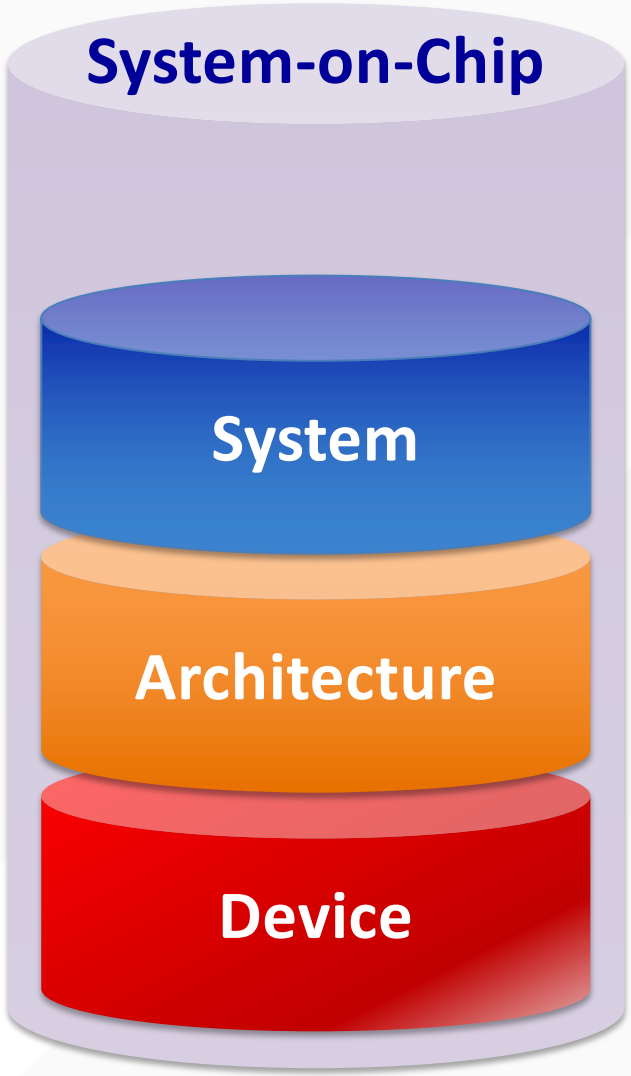
- S.Gonugondla, M.Kang et al.,
  [JSSC 18Sep] [ISSCC 18]





- M.Kang et al.,
  [Book, Springer 2020]
  [Proceedings of IEEE, 2020]

Vertically-integrated VLSI Information Processing (VVIP) Lab



Research goal:

- End-to-end VLSI implementation for machine learning and signal processing algorithms

- Vertically-integrated research across circuit, architecture, system with both CMOS and emerging devices

https://www.ucsdvvip.com/

Students' Background ?

Machine Learning in Daily Life

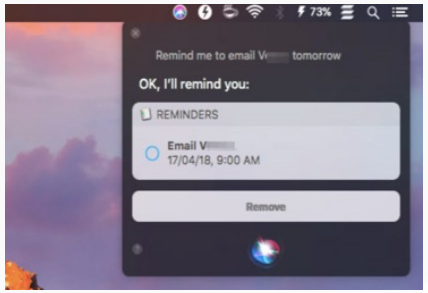
SPAM(1)

6:30 AM

7:00 AM

7:30 AM

8:00 AM



8:30 AM

To:
cc:
Thank you for your time

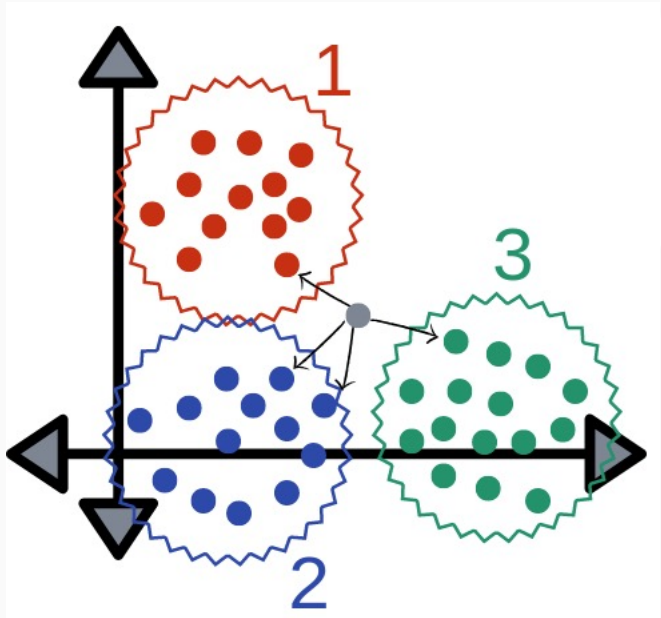
9:00 AM

What is Machine Learning ?

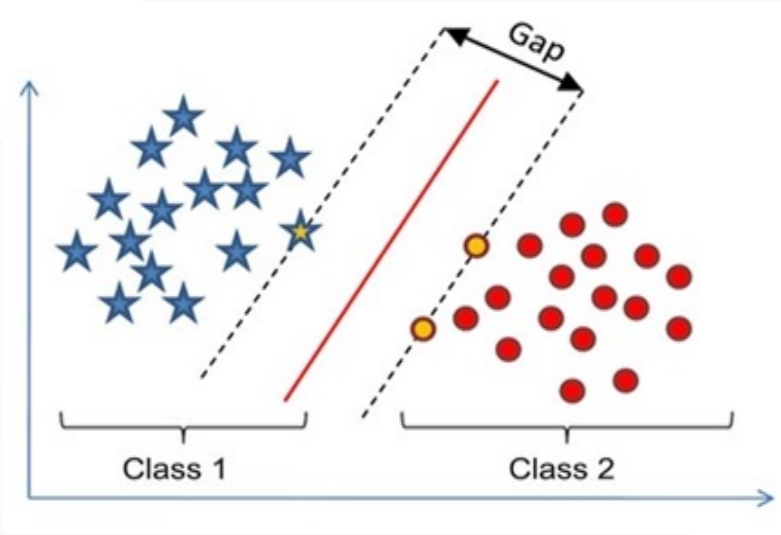
- Definition of **Machine learning**

  - the study of computer algorithms that improve automatically through experience and by the use of data by Wikipedia

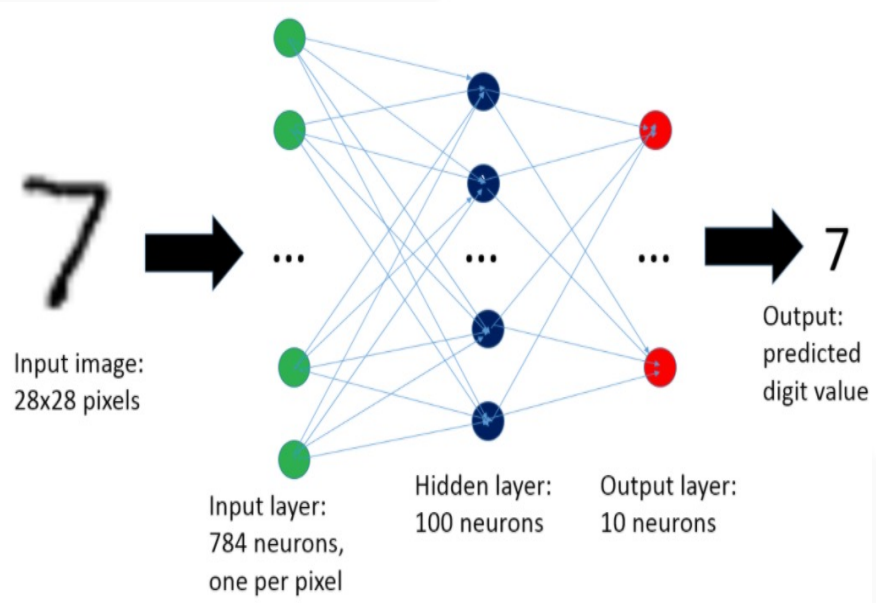
  - field of study that gives computers the ability to learn without being explicitly programmed by Arthur Samuel, 1959



K-nearest neighbor



support vector machine

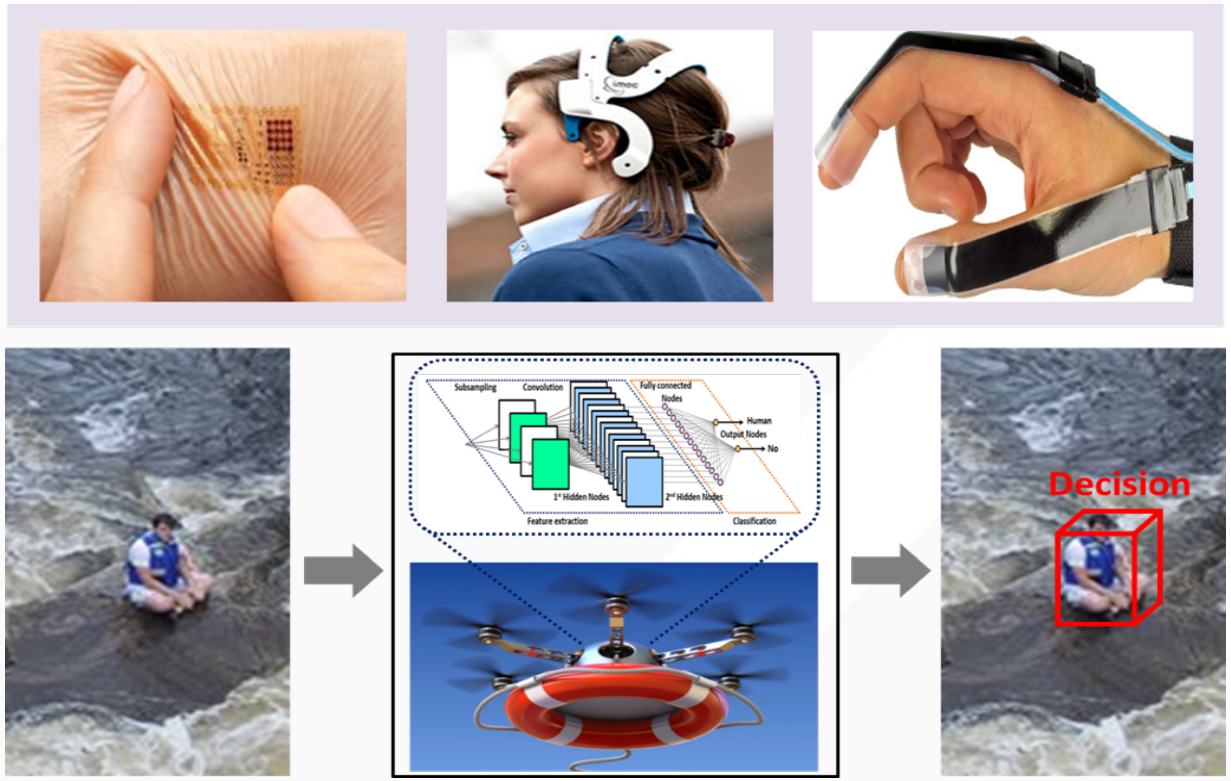


neural network

figures from: https://machinelearninghd.com, https://medium.com/, https://db-blog.web.cern.ch/

Machine Learning under Resource Constraints

Liquid cooling system in data center



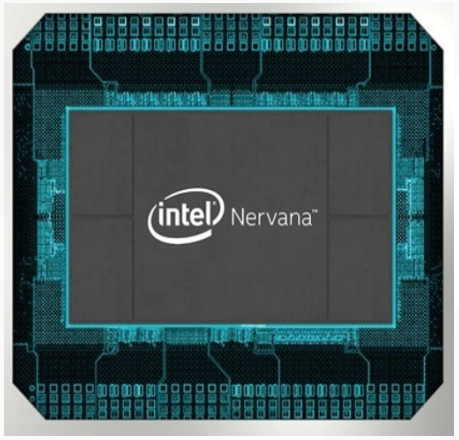
Decision making at the Edge

Data & computation ↑ + resource budget ↓

figures from https://www.datacenterknowledge.com

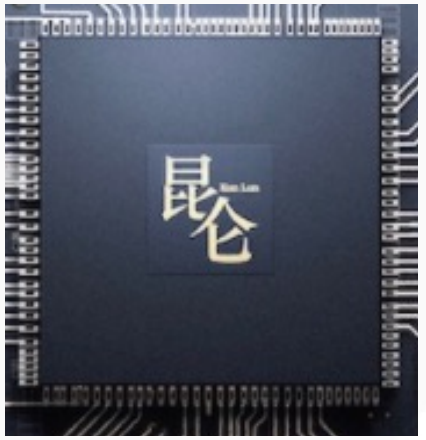
Machine Learning (ML) Hardware ASIC

Google TPU



INTEL Nervana

IBM TrueNorth



Alibaba

Snapdragon

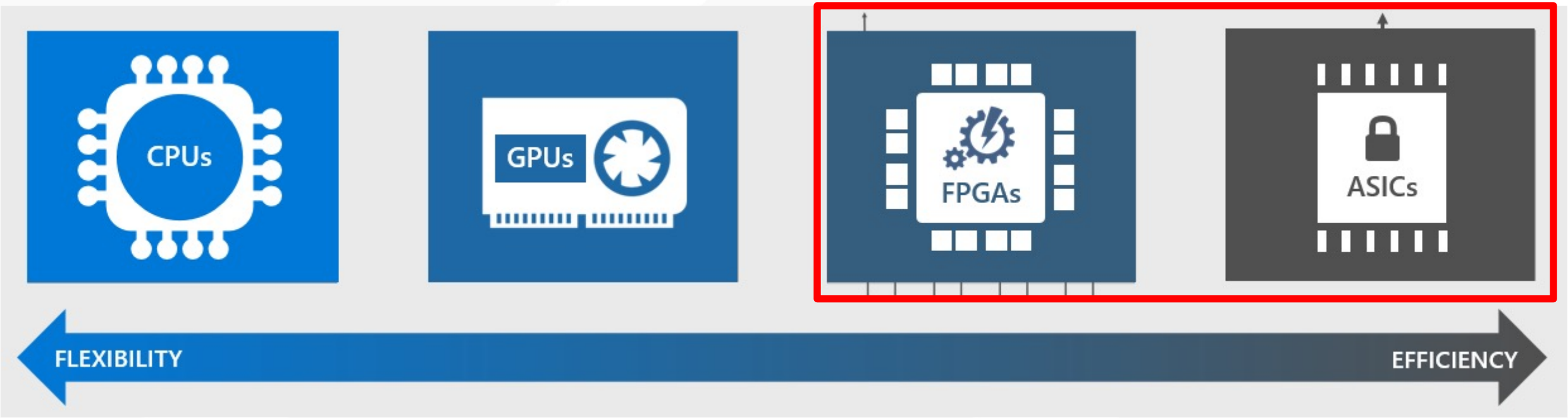
~75W

~10W

>> 10-100mW, still power hungry

- Application specific IC (ASIC) implementations for DNN
- Targeting aggressive energy and delay efficiency for DNN algorithms

Why we are learning this course ? Trade-offs in Hardware



GOPS / W

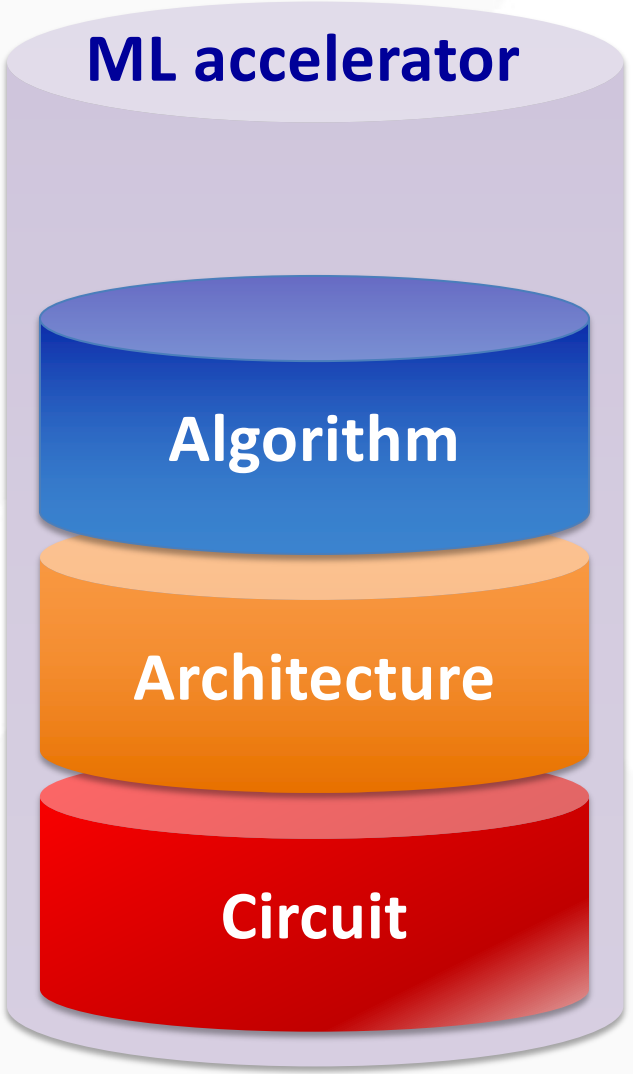
TOPS / W

Figure from http://cryptopress.online

Course Overview

Goal of this course:

- provide the vertical learning across algorithm, architecture, and circuit layers to design energy-efficient and high throughput machine learning accelerator
- provide “hand-on” experience of training your network models, maps on hardware, verify the functionality, and measure the performance
- introduces a state-of art algorithm, architecture, and circuit techniques in ML domain



Core Topic

Algorithm topics:

- Deep Neural Networks
- Non DNN algorithms
  (Xgboost, KNN, HD Neuromorphic computing, …)
- Quantization
- Pruning
- Compression
- Computer vision
- Natural language processing
- Pytorch programing

Hardware topics:

- Arithmetic design
- Sparsity-aware design
- Algorithm to hardware mapping
- AI architectures
  (2D systolic array, 1D vector processor)
- Other advanced technology
  (In-memory computing, analog processor,
  approximate computing)
- Verilog design / verification
- Synthesis with Quarters Prime

Is this course a good choice for you ? (Prerequisite)

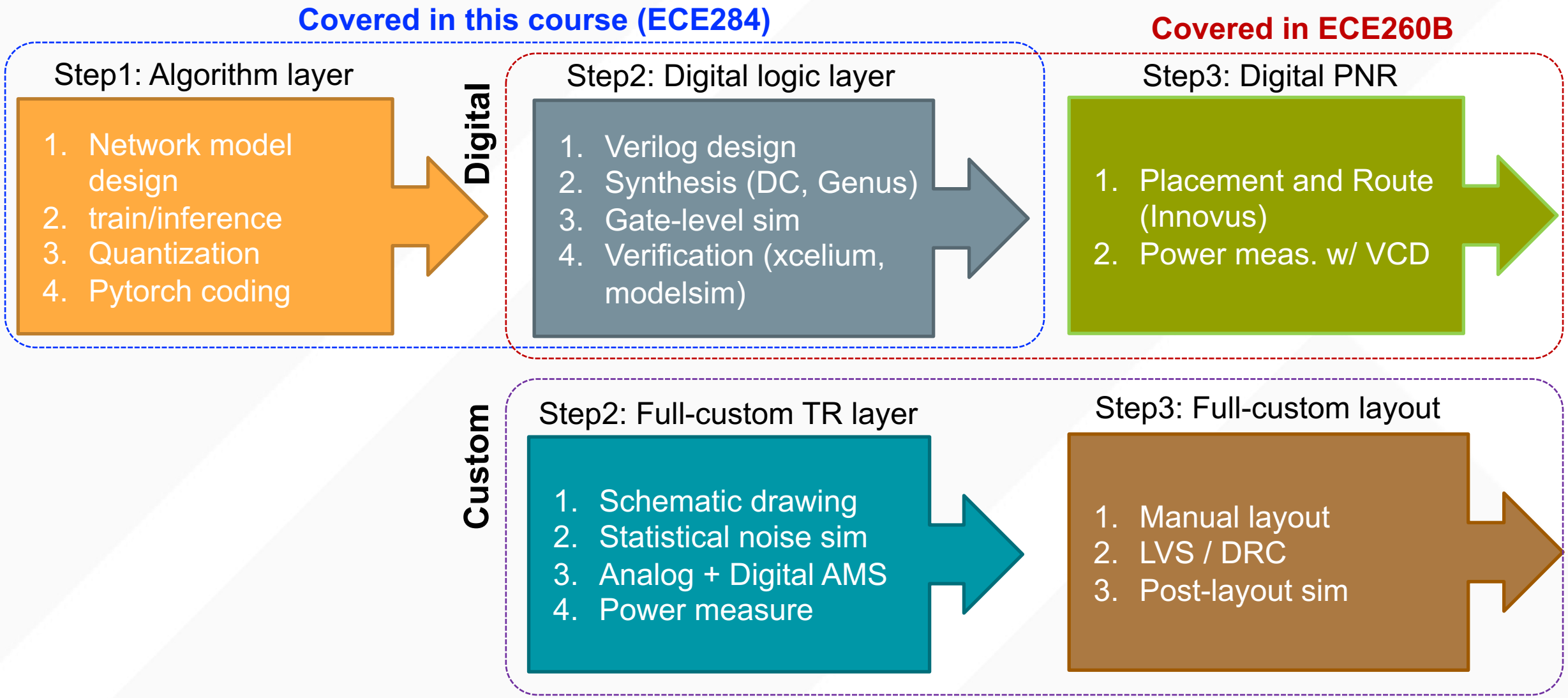
Good idea if you:

- want to work in the ML accelerator industry in the future
- want to learn how ML algorithm is mapped on hardware
- are hardware-focused researcher, but want to explore ML to apply in your area
- algorithm designer, but want to have better understanding how the accelerator works
- want to learn ASIC / accelerator in general
- explore this area for your future research or further degree

Bad idea if you:

- are looking for deep theory of ML
- want to learn pure algorithms in ML
- expected laid-back seminar course
- don’t know digital logic design and verilog program coding at all (ECE111)
- don’t know either C++ or Python

Where is this course in ECE Curriculum ?



Step1: Algorithm Layer (Pytorch Example)

```python
import torch
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120)  # 5*5 from image dimension
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square, you can specify with a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = torch.flatten(x, 1) # flatten all dimensions except the batch dimension
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Step2: Digital Logic Design (Verilog Behavioral Design)

```
module signed_mult (out, clk, a, b);

        output          [15:0]  out;
        input                   clk;
        input   signed  [7:0]   a;
        input   signed  [7:0]   b;

        reg     signed  [7:0]   a_reg;
        reg     signed  [7:0]   b_reg;
        reg     signed  [15:0]  out;

        wire    signed  [15:0]  mult_out;

        assign mult_out = a_reg * b_reg;

        always@(posedge clk)
        begin
                a_reg <= a;
                b_reg <= b;
                out <= mult_out;
        end

endmodule
```

Step2: Synthesis (Gate-level Netlist)

```
module carry(input  a, b, c,
             output cout)

    wire   x, y, z;

    and g1(x, a, b);
    and g2(y, a, c);
    and g3(z, b, c);
    or  g4(cout, x, y, z);
endmodule
```

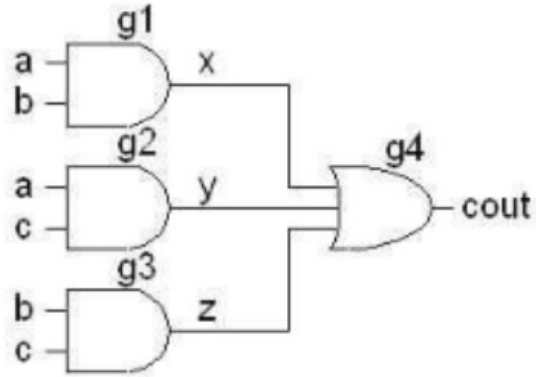


figure: https://www.fpgakey.com/wiki/

Step3: Digital Placement and Route (PnR)

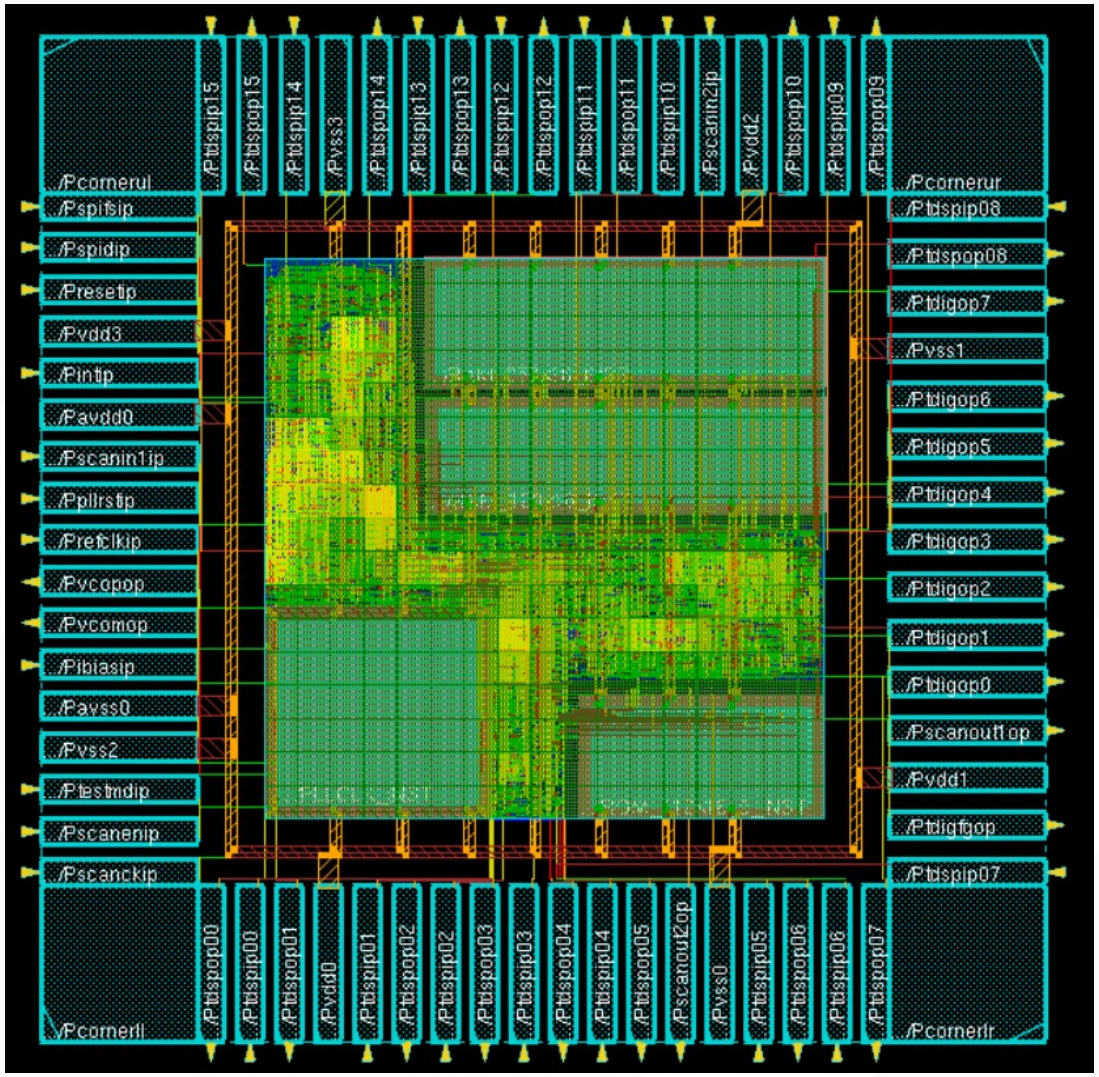
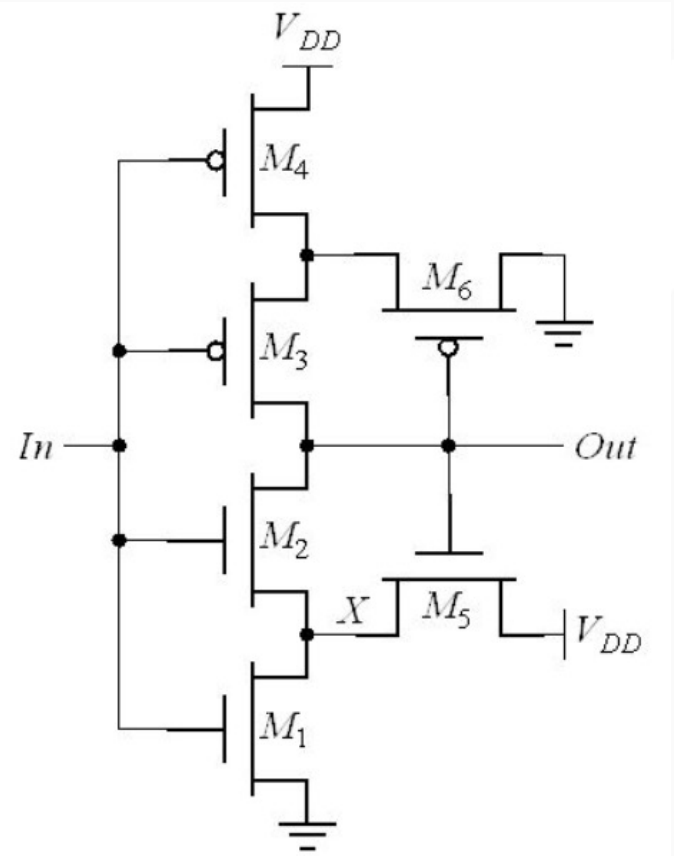


figure: www.cadence.com

Step2: Full-Custom TR-level Design (Cadence Virtuoso)



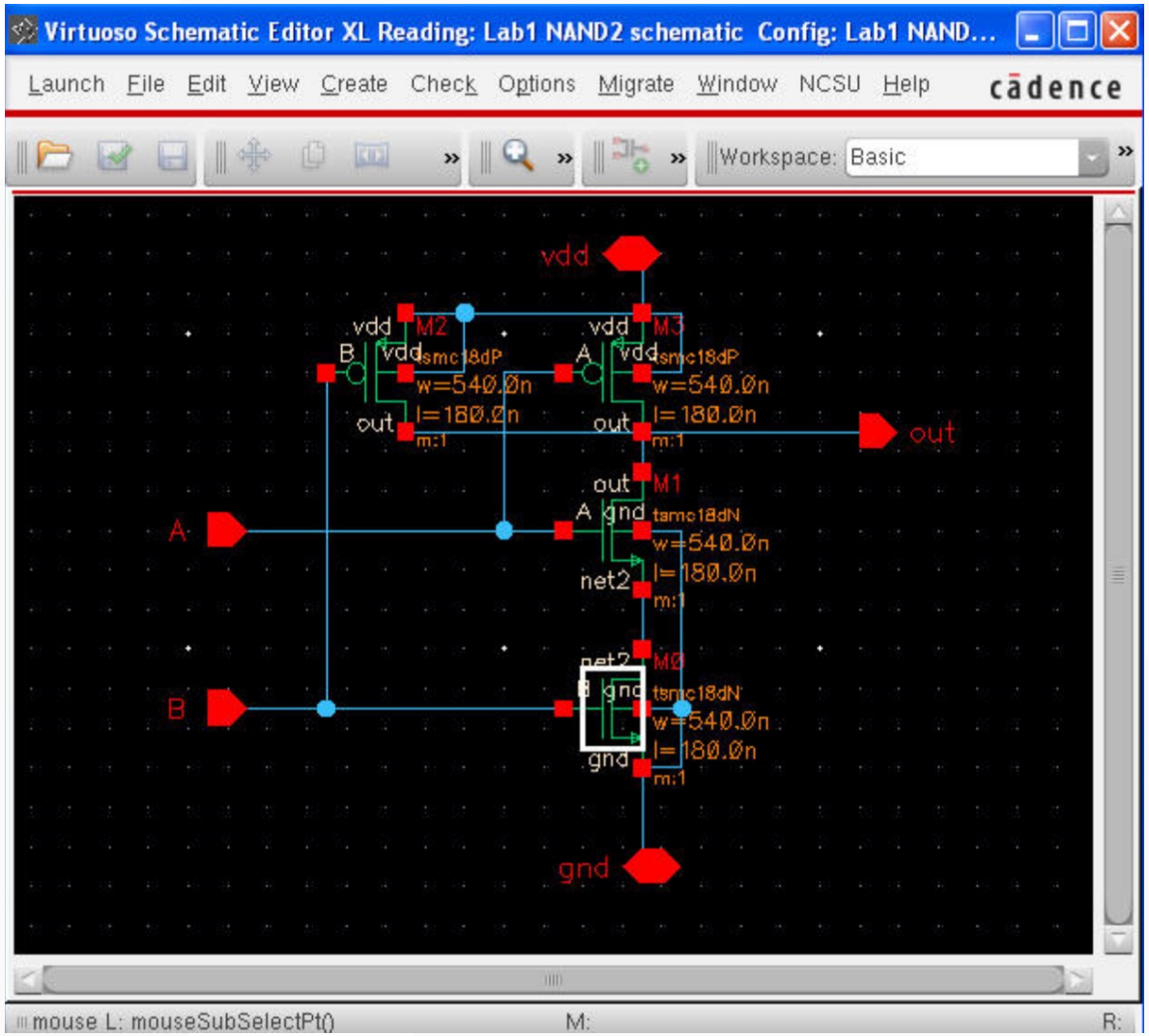


figure: www.engrclasses.pitt.edu

Step3: Full-Custom Layout (Cadence Virtuoso)

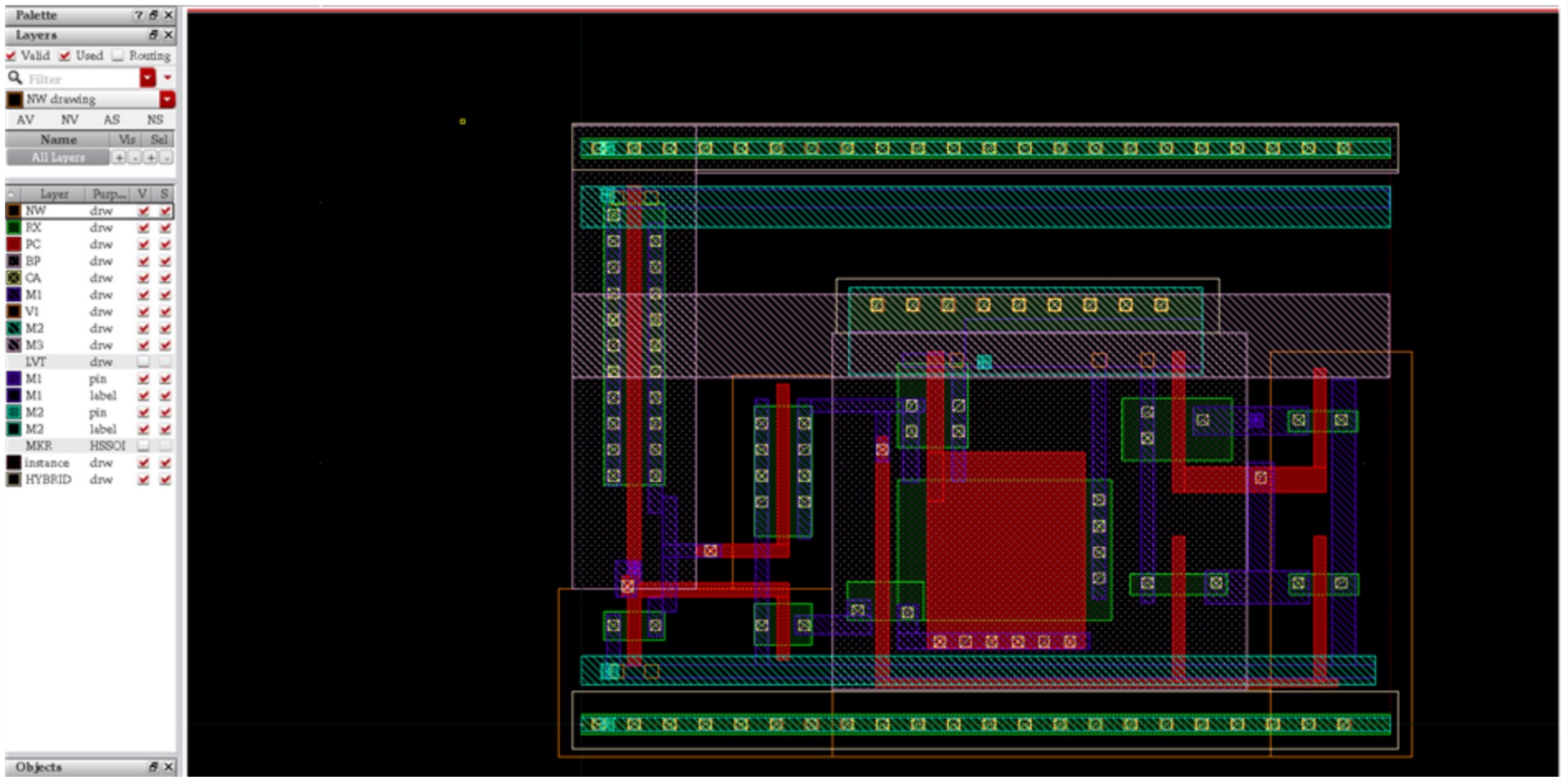
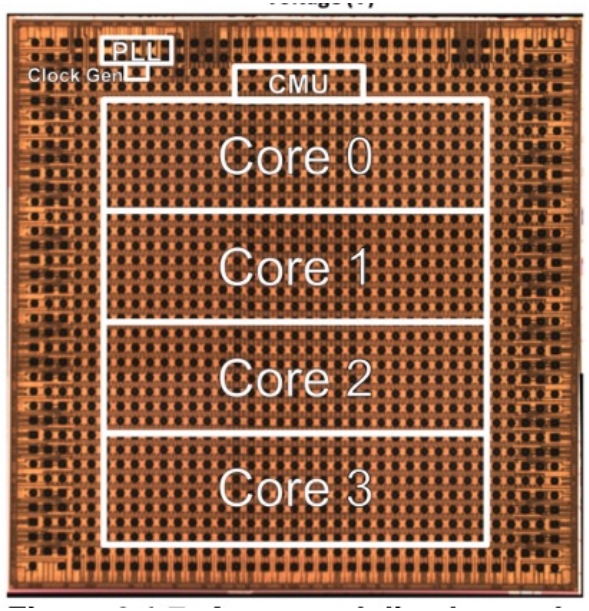
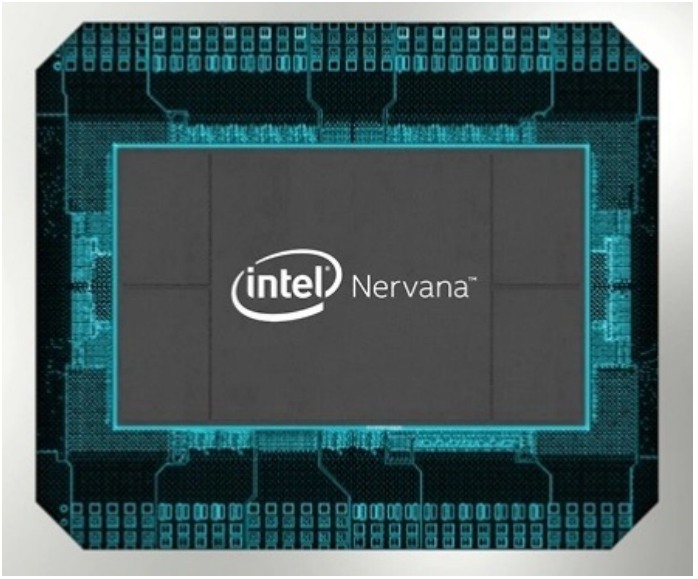


figure: https://community.cadence.com/

Weekly Plan

2D Systolic Architecture

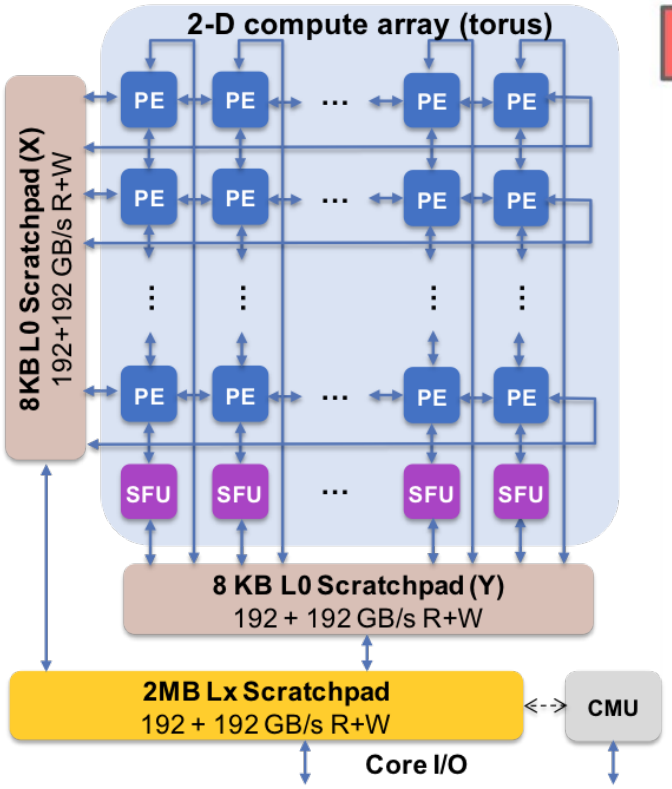




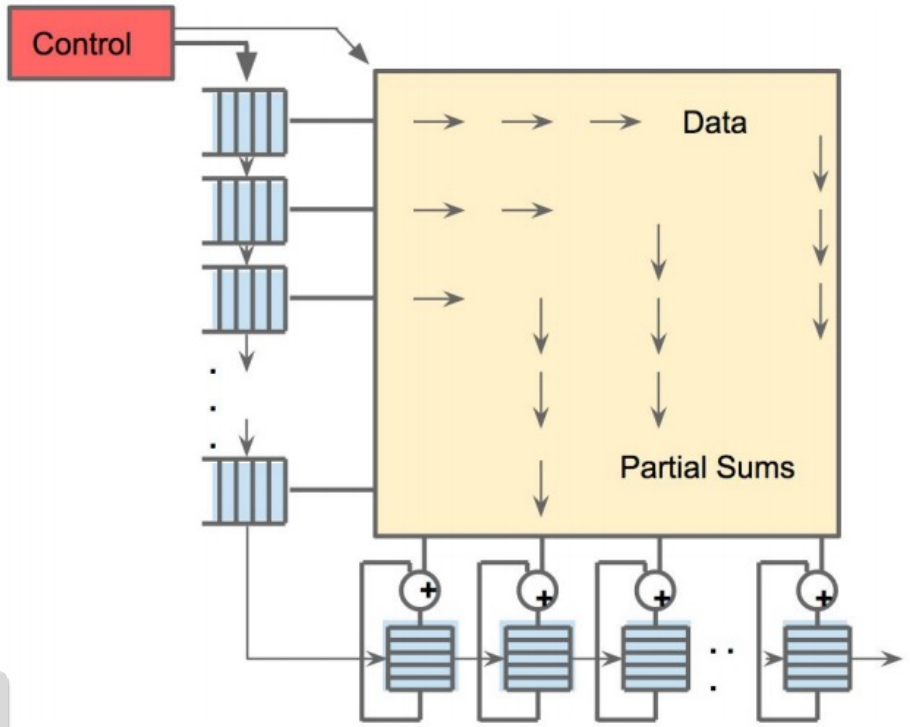
IBM Rapid **Google TPU** **Intel Nervana**

J.Oh, “A 3.0 TFLOPS 0.62V Scalable Processor Core for High Compute Utilization AI Training and Inference“
https://arxiv.org/pdf/1704.04760.pdf
https://fuse.wikichip.org/news/3270/intel-axes-nervana-just-two-months-after-launch/

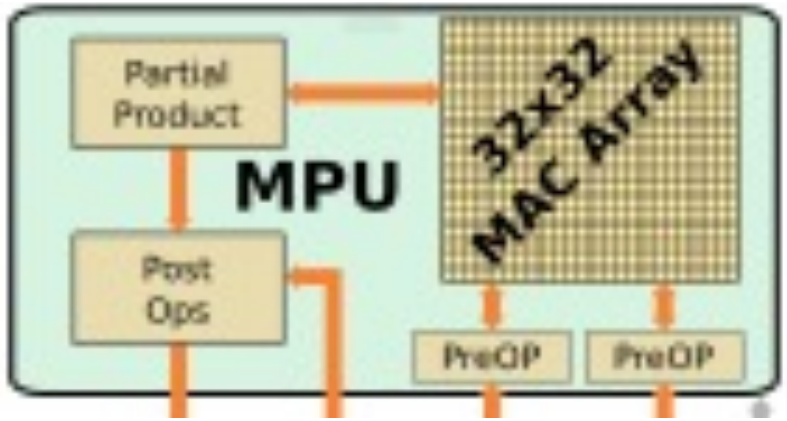
2D Systolic Architecture



IBM Rapid



Google TPU



Intel Nervana

J.Oh, “A 3.0 TFLOPS 0.62V Scalable Processor Core for High Compute Utilization AI Training and Inference“
https://arxiv.org/pdf/1704.04760.pdf
https://fuse.wikichip.org/news/3270/intel-axes-nervana-just-two-months-after-launch/

Course Logistics

- In-person lecture
- Location: Jacobs Hall (EBU1) 2315
- Audio and screen recorded and posted (in Canvas)

  (You can request editing for video clips)

  · Recording quality not guaranteed

  · Black board drawing will be missing
- Course website: Canvas https://canvas.ucsd.edu/courses/29661 (No Piazza)

  · Check it once a day (your responsibility)

Health Guideline

- What if you are not well ?
  - Do not attend in-person lecture, midterm, and final presentation
  - Will be replaced by remote
  - Same for instructor and TA

Instructor Contact

- Instructor office hours: Tuesday 2 – 2:50pm
- Location: Jacobs Jall (EBU1) 6406
- Contact: only through ece284ucsd@gmail.com

  (Other email account not guaranteed to be read on time)
- [PROFESSOR ONLY] in the subject line, if you want to communicate directly with an instructor.

TA Introduction

- Abhiram Medisetti
- Office hour: 2 – 2:50 Wednesday
- Location: Jacobs Jall (EBU1) 1511

Grading

Absolute grading

1. HW 30%
2. Midterm 30%
3. Final project 40%

HW (30%)

- Full score for each HW can be different

  (i.e., impact factor of each HW is different)

- Submit through Canvas

- Deadline: by 1pm on Monday

- Solution will be posted on Monday

- Not a teamwork, but discussion and Canvas Discussion tab allowed

- Copying others is strictly prohibited. Submit your own homework

- Note: GPU is limited (early bird gets the GPU)

Project (40%)

- Will be given in week 4 or 5.
- Due date: Nov 29 1pm through canvas
- Note: No more questions will be answered after two days before the deadline
- Team (tentatively 2 students) formation needs to be submitted by week3 Friday.

  Or, will be connected randomly.
- Presentation on Nov 29 & Dec 1 during class, If necessary, Dec. 10 additionally.
- Other's participation required during the presentation (for grading)

Project (40%)

- What if team members have trouble ?

- If team members are separated, separated members can form a new team

- Otherwise (single person for a team), 10% score benefit is given

Office for Students with Disabilities (OSD) Accommodation

- Send me Authorization For Accommodations (AFA) letter immediately
- Make sure my acknowledgement email from me

In-person and Online Etiquette

- 30+ students, but one instructor + 1 TAs
- Difficulty to debug each student's coding or technical issue
- Questions: clearly + slowly
- Respectful to TAs and other students
- Discussion forum – check previous questions, no complaints but please email to me
- Right before HW / project deadline, answer will be difficult, so ask the questions earlier.

Course Materials

- No textbook
- Course github: https://github.com/VVIPLab/ece284fa2021
- UCSD datahub: https://datahub.ucsd.edu/Links to an external site.

  (For connection outside campus, UCSD VPN needs to be on.)
- UCSD VPN: https://blink.ucsd.edu/technology/network/connections/off-campus/VPN/Links to an external site.
- UCSD Linuxcloud: https://linuxcloud.ucsd.edu/Links to an external site.
- Quartus prime: either UCSD Linuxcloud or personal labtop

Work Environment Test

- UCSD datahub:
  - Check you have ECE284 “GPU” setup
  - Once you log-in, open new Jupyter Notebook by selecting “New-Python3” tab
  - Type as follows in the cell, and run by typing “shift + enter”:

```
import torch
print(torch.cuda.is_available())
```

  - Above command should output “True”

- UCSD Linuxcloud:
  - Once you log-in, type command “iverilog”
  - It should give error message “iverilog: no source files”
  - Type command “gtkwave”: it should open waveform viewer

- Please send an email if above setup is not available

Software Download and Setup in Local Machine

1. Jupyter notebook tutorial (https://www.youtube.com/watch?v=jZ952vChhuI)

2. Icarus Installation (Manual: http://www.referencedesigner.com/tutorials/verilog/verilog_02.php)

3. GTKwave Installation (http://gtkwave.sourceforge.net/)

4. Quarters Prime Installation:

(http://cwcserv.ucsd.edu/~billlin/classes/ECE111/quartus_modelsim_tutorial_4_1_18/quartus_modelsim_tutorial.html)