Multi-layer Perceptron Design for Human Activity Recognition

Fall 2021 – ECE Design Project

Important: Revised on November 7th

Revised Slides:

- **Slide 5:** Weight w01 changes from -1 to -2 (see the slide)
- **Slides 8-12:** Since the weight are fixed, the output of the RELU Mux is guaranteed to be less than or equal to 2. Therefore, you do not have to implement the saturation function (shift by 1). That is, this change will eliminate one block in your design.

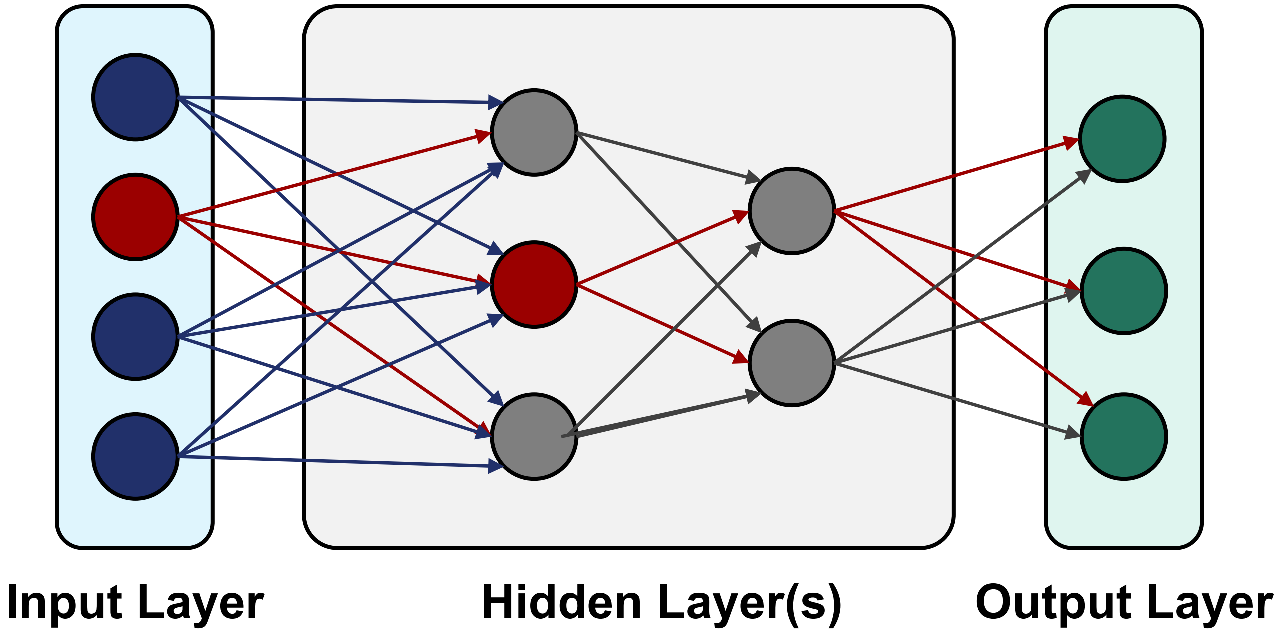
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Motivation for the Project

- **Deep learning is currently perhaps the most popular application area**
  - From applications to algorithms and hardware design
- **One can follow a modular hardware design methodology**
  - Start from a basic building block a neuron
  - Form a hidden layer by instantiating neurons
  - Instantiate multiple hidden layers by reusing one hidden layer template
    - e.g., a parameterized number of layers can be pipelined
- **The basic building block itself involves fundamental components**
  - Adder
  - Shifter
  - Multiplier (simplified version)
  - Multiplexer

Motivation for the Project (cont')

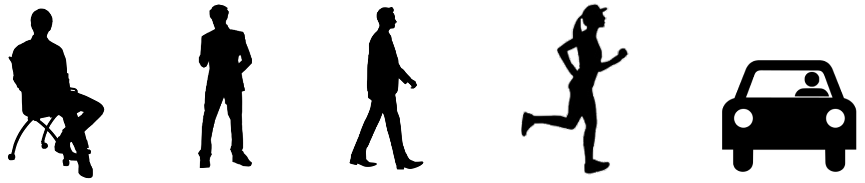
- **An opportunity for you to**
  - Design the basic building block of a fundamental architecture
  - Practice design and layout skills on fundamental datapath components (e.g., adder, shifter)
  - Learn how to contribute towards a system-level goal



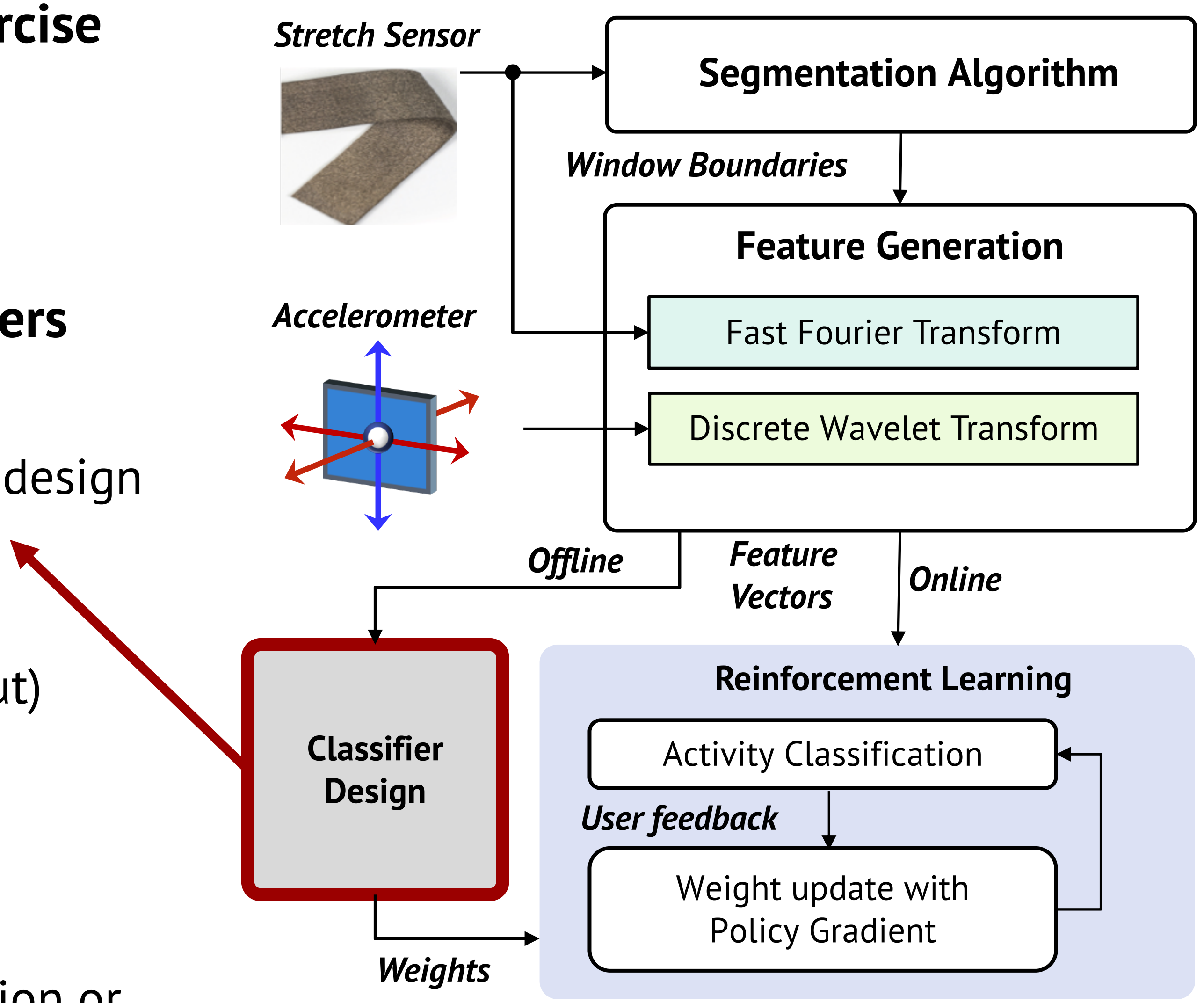
- **Input nodes**
  - Simply pass the information from the environment without processing
  - Each input can be a pixel in an image, or a single sensor datapoint
- **Hidden layer(s)**
  - Perform computations
  - Inner product of inputs and weights, followed by an activation function
  - Transfer the information towards the ouput
- **Output nodes**
  - Convert the information from the network back to the environment (e.g., a classification result)

System—level View: Human Activity Recognition (HAR)

- **Identify activities, such as walk, sit, and exercise**

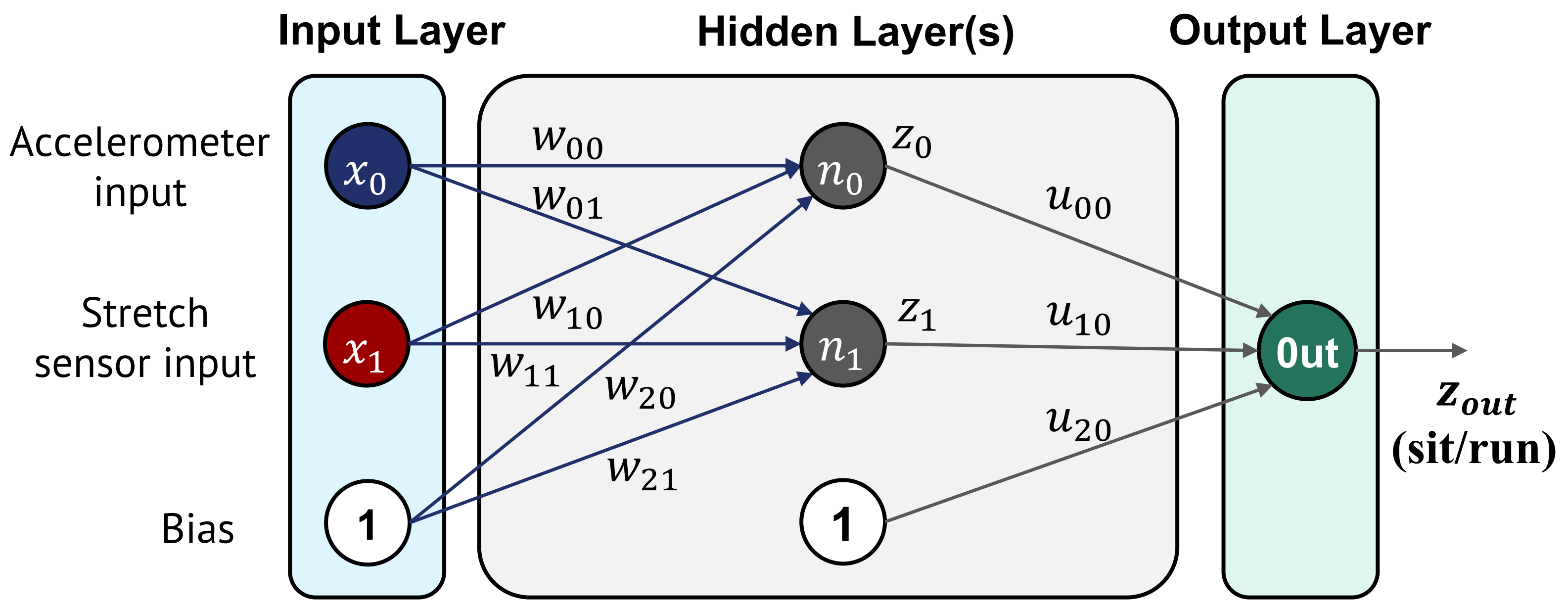


- **First step to solutions for movement disorders**
- **Your task in this project**
  - A *_simplified_* version of the activity classifier design
  - You will focus on only inference (forward pass), i.e., the weights are given to you
  - It will classify two activities (i.e., binary output)
    - Sit, Run
  - The inputs* are
    - 1D accelerometer measurement
    - Stretch sensor attached on a knee sleeve
  - *You do not need to worry about the application or input; everything you need will be given to you



[1] Bhat, Ganapati, et al. "Online human activity recognition using low-power wearable devices." _Intl. Conf. on Computer-Aided Design (ICCAD)_. IEEE, 2018.

Multi-layer Perceptron Operation and Weights



Computations

At	Step 1	Step 2 (Neuron output)
n_0	$y_0 = x_0 w_{00} + x_1 w_{10} + w_{20}$	$z_0 = \max(0, y_0)$
n_1	$y_1 = x_0 w_{01} + x_1 w_{11} + w_{21}$	$z_1 = \max(0, y_1)$
Out	$y_{out} = z_0 u_{00} + z_1 u_{10} + u_{20}$	$z_{out} = \max(0, y_{out})$

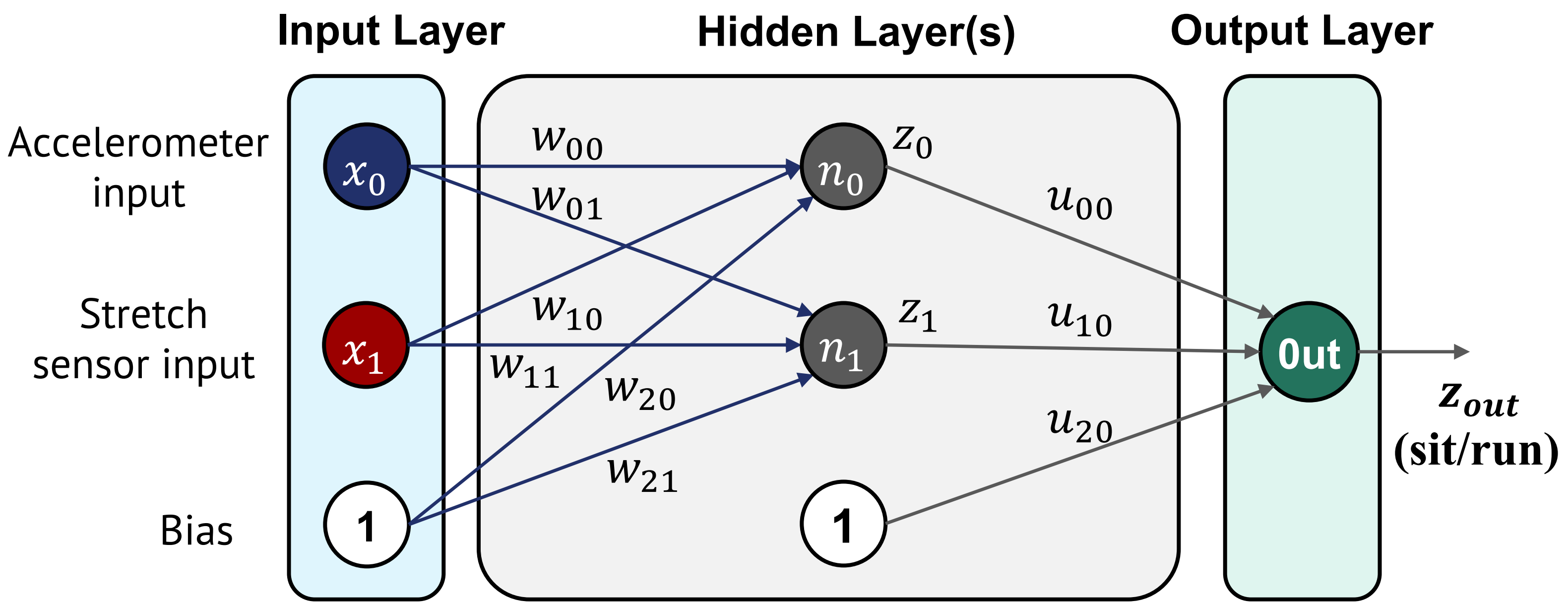
Inner product of the input and weights **Relu Activation**

MLP Network Weights

- Standard training algorithms, such as stochastic gradient descent (SGD), find the weights for a given supervised learning algorithm
- For example, suppose
  - $z_{out} > 1$ implies the activity is SIT
  - $z_{out} \leq 1$ implies the activity is RUN
- Training with known sensor data and labels will determine the weights, which will be given to you
- **Assume that the weight are as follows:**
  - $w_{00} = -1$, ~~$w_{01} = -1$~~
  - $w_{01} = -2$
  - $w_{10} = w_{11} = u_{00} = u_{10} = 1$
  - $w_{20} = w_{21} = u_{20} = -1$

Revision

Multi-layer Perceptron: A simple illustration - 1



Computations

At	Step 1	Step 2 (Neuron output)
n_0	$y_0 = x_0 w_{00} + x_1 w_{10} + w_{20}$	$z_0 = \max(0, y_0)$
n_1	$y_1 = x_0 w_{01} + x_1 w_{11} + w_{21}$	$z_1 = \max(0, y_1)$
Out	$y_{out} = z_0 u_{00} + z_1 u_{10} + u_{20}$	$z_{out} = \max(0, y_{out})$

Inner product of the input and weights

Relu Activation

Example Input/Output

Case 1 Inputs

- $x_0 = 0$ (no motion)
- $x_1 = 3$ (sensor is stretched)

Hidden layer

- $y_0 = 3w_{10} + w_{20} = 2 \Rightarrow \boldsymbol{z_0} = \mathbf{2}$
- $y_1 = 3w_{11} + w_{21} = 2 \Rightarrow \boldsymbol{z_1} = \mathbf{2}$

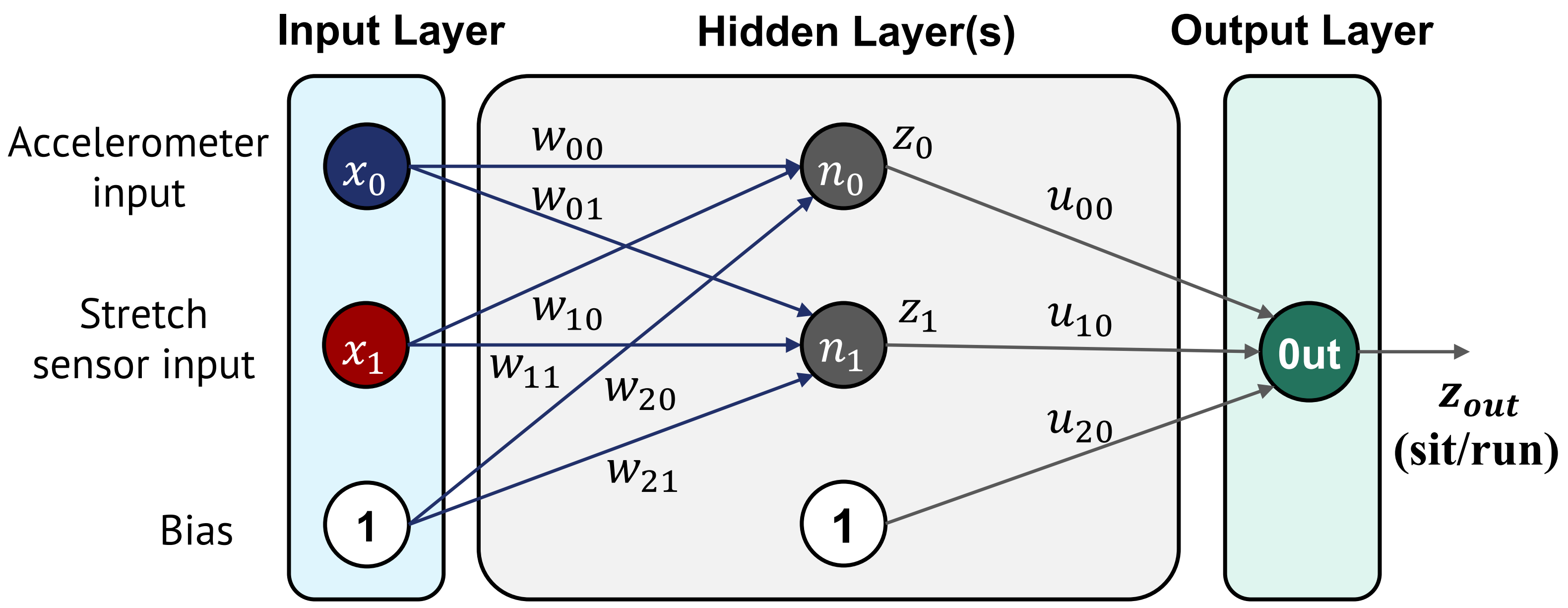
Output

- $y_{out} = z_0 u_{00} + z_1 u_{10} + u_{20} = 3$

$\Rightarrow z_{out} = 3$

That is, the output $z_{out} > 1$ implies "SIT"

Multi-layer Perceptron: A simple illustration - 2



Computations

At	Step 1	Step 2 (Neuron output)
n_0	$y_0 = x_0 w_{00} + x_1 w_{10} + w_{20}$	$z_0 = \max(0, y_0)$
n_1	$y_1 = x_0 w_{01} + x_1 w_{11} + w_{21}$	$z_1 = \max(0, y_1)$
Out	$y_{out} = z_0 u_{00} + z_1 u_{10} + u_{20}$	$z_{out} = \max(0, y_{out})$

Inner product of the input and weights — **Relu Activation**

Example Input/Output

Case 2 Inputs

- $x_0 = 2$ (significant motion)
- $x_1 = 0$ (sensor is NOT stretched)

Hidden layer

- $y_0 = 2w_{00} + w_{20} = -3 \Rightarrow \mathbf{z_0 = 0}$
- $y_1 = 2w_{01} + w_{21} = \mathbf{-5} \Rightarrow \mathbf{z_1 = 0}$

Output

- $y_{out} = z_0 u_{00} + z_1 u_{10} + u_{20} = -1$

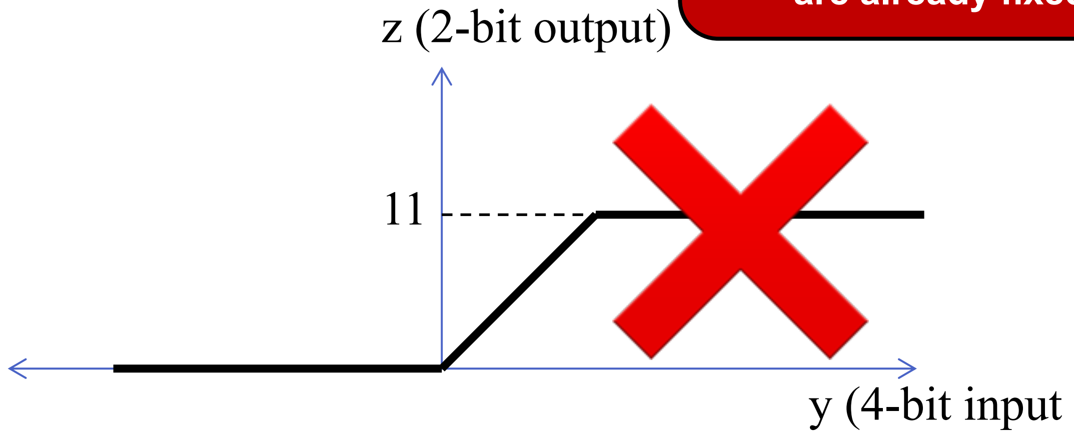
$\Rightarrow z_{out} = 0$

That is, the output $z_{out} < 1$ implies "RUN"

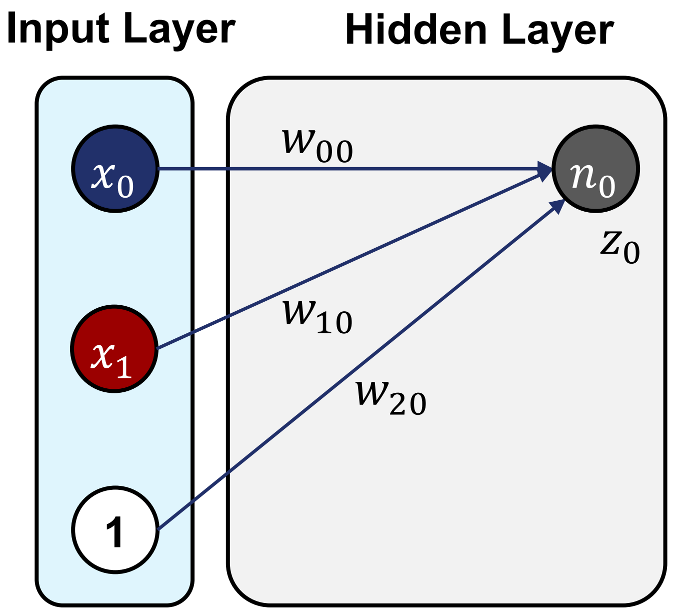
Simplifying Assumptions

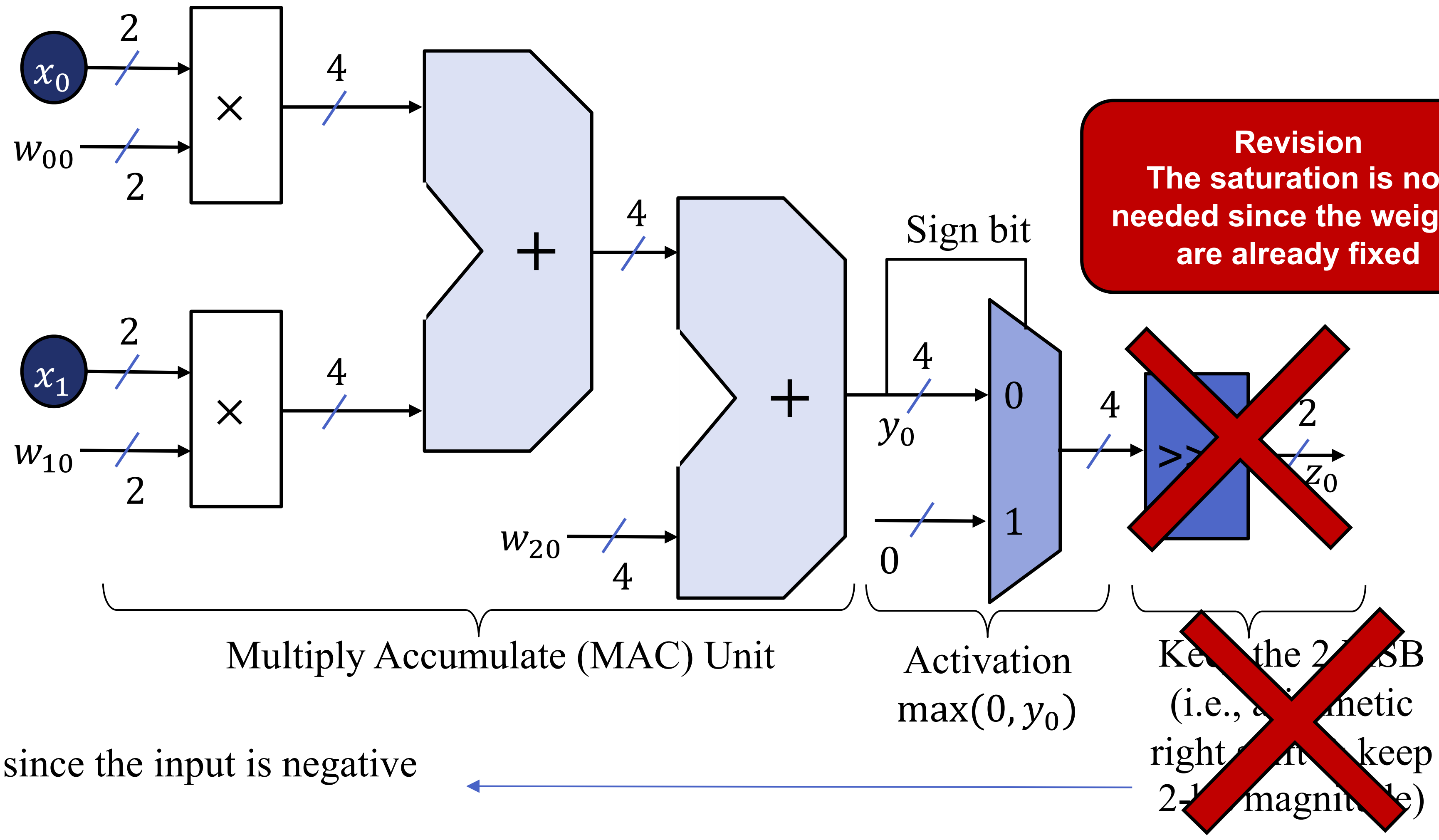
- **Inputs**
  - Assume that the inputs are normalized to the interval [0,1], which is a common practice
  - Further, assume that each input is discretized to 2 unsigned bits: ***2'b00, 2'b01, 2'b10, 2'b11***
  - The last assumption is made to avoid a multiplier design
    - Multiply by 2 is arithmetic left shift; multiply by 3 can be implemented by left shift and addition
- **Weights**
  - Assume the weights are 2 bit signed numbers: 2'b10 = -2; 2'b11 = -1; 2'b00 = -0; 2'b01 = 1
- **You can hardcode the weights into your design**
  - In practice, the weight should be loaded from memory
- **Activation function output z = f(y)**
  - z = 0: If the input is negative
  - z = y: If the input is positive~~*~~
  - ~~Keep the most significant 2-bits (saturate) to avoid overflow~~

Revision
The saturation is not needed since the weights are already fixed



Example Datapath for a Neuron





x_0
w_{00}
x_1
w_{10}
w_{20}
2
4
×
+
Sign bit
0
1
y_0
>>
z_0
Multiply Accumulate (MAC) Unit
Activation $\max(0, y_0)$
Keep the 2 LSB (i.e., arithmetic right shift, keep 2-bit magnitude)

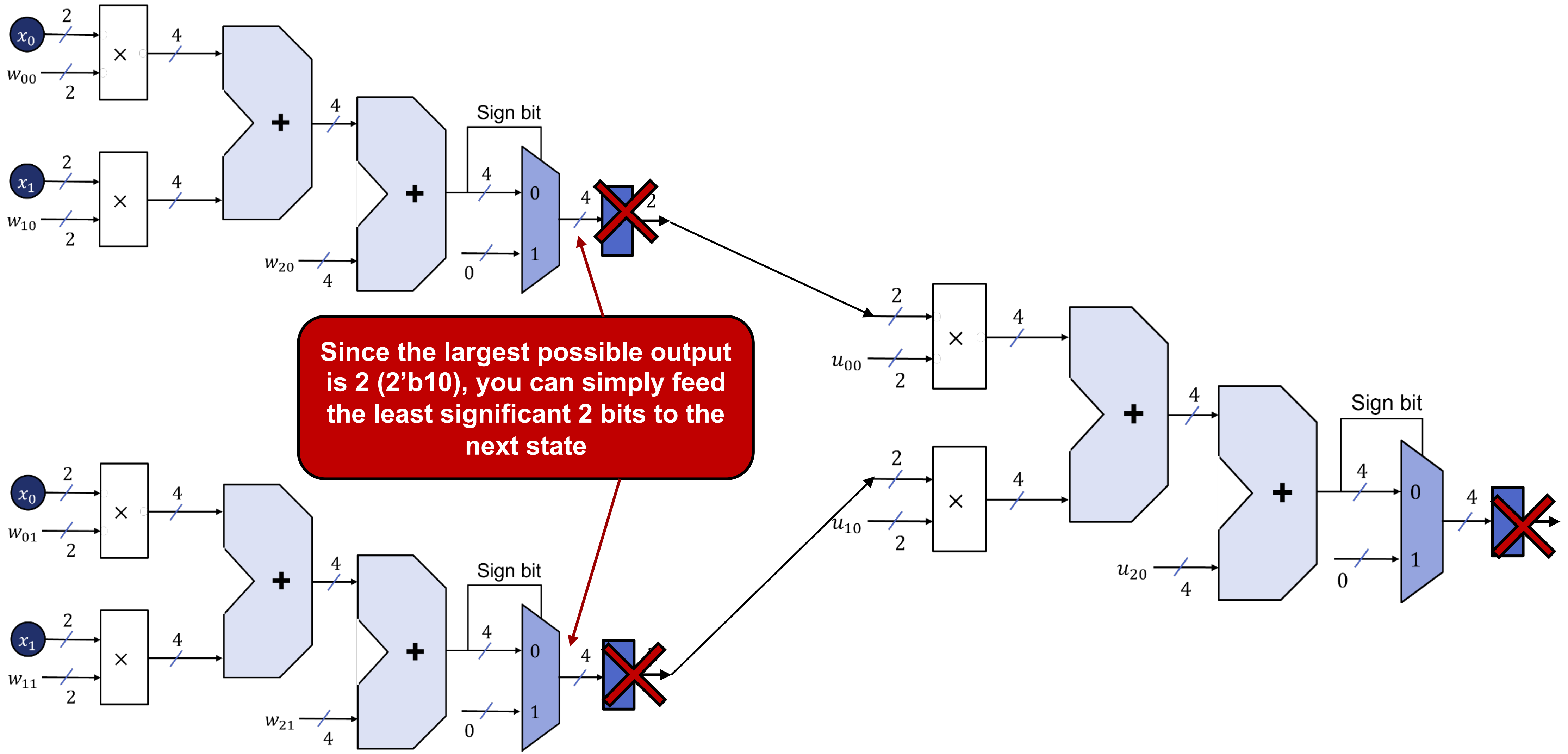
Revision
The saturation is not needed since the weights are already fixed

New Examples:

$y_0 = 1XXX \rightarrow z_0 = 0$ since the input is negative

$y_0 = 0XXX \rightarrow z_0 = XXX$ since the input is positive

Datapath of the full MLP



Since the largest possible output is 2 (2'b10), you can simply feed the least significant 2 bits to the next state

A Quick Note on Multiplication

- **We did not cover multipliers, but you know shift left and addition!**
  – In fact, multiplication consist of shift and repetitive additions

x_i	Product	Comment
00	0	Output is 0
01	w	Output is same as the weight
10	2w	Arithmetic shift left
11	3w	Arithmetic shift left and add (2w+w).

Overflow could have happened, but we will use 4 bits for the products to avoid it

Since the weights are given (1, -1, -2), you only need shift and 2's complement operations

Conversion from 4-bit to 2-bit

- **The final output of a neuron will only have 2bits**
  - Drop the MSB of the output of the activation and right shift by 1

Input	Output
~~0111~~	~~11~~
~~0110~~	~~11~~
[illegible]	[illegible]
[illegible]	~~10~~
[illegible]	[illegible]
[illegible]	[illegible]
~~0001~~	~~00~~
~~0000~~	~~00~~
~~1XXX~~	~~00~~

Since we dropped the saturation, this step is not needed anymore. You can simply retain the Least Significant 2 bits (as mentioned before)

Input	Output
0111	11
0110	10
0101	01
0100	00
0011	11
0010	10
0001	01
0000	00
1XXX	00

Project Specifications

1. **Each neuron of MLP is homogeneous (MAC+activation)**
   - Implement a neuron using the recommended data path
2. **Design a single layer of MLP**
   - Instantiate as many neurons as needed (in our case 2) and connect the input
   - One can easily design taller layers
3. **Implement the output layer using the same neuron design**

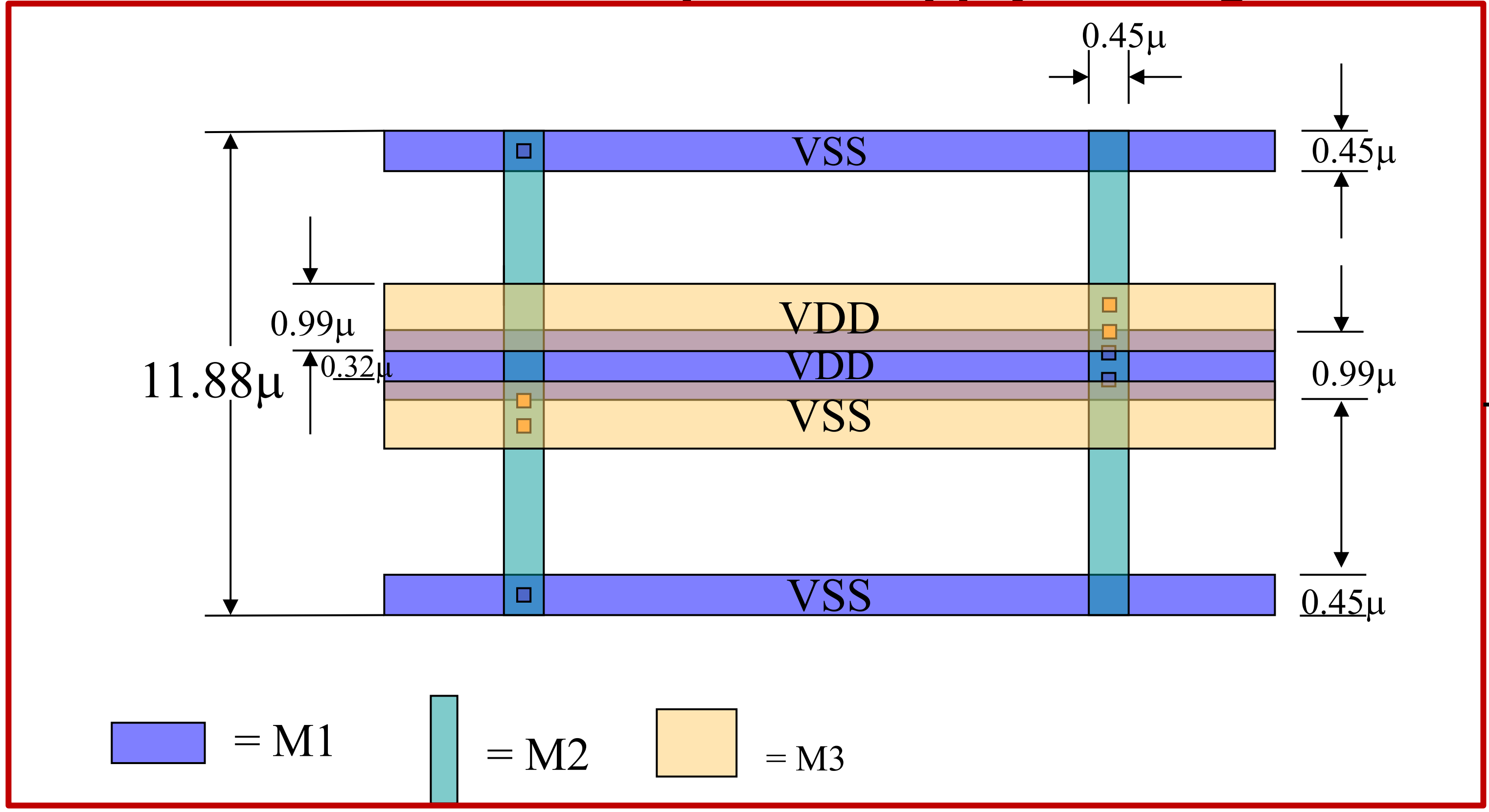
- **In general, one can replicate the hidden layers**
- **One can use 8 or a larger number of bits (with a proper MAC design)**
  - 8-bit quantized weights/activations provide >90% accuracy for HAR

Milestones & Expected Outcome

- **The project involves all phases of design:**
  - Schematic design and capture Cadence
  - Pre-layout simulation with
  - Layout with Cadence Virtuoso (LVS/DRC clean)
  - Parasitic extraction and resimulation/characterization with extracted parasitics
- **Milestone-1: Designing a Neuron**
  - Deadline: November 23rd
- **Final: Designing the complete MLP**
- **Final design will be evaluated based on operating frequency**

Optimization Idea (Bit Slicing Technique)

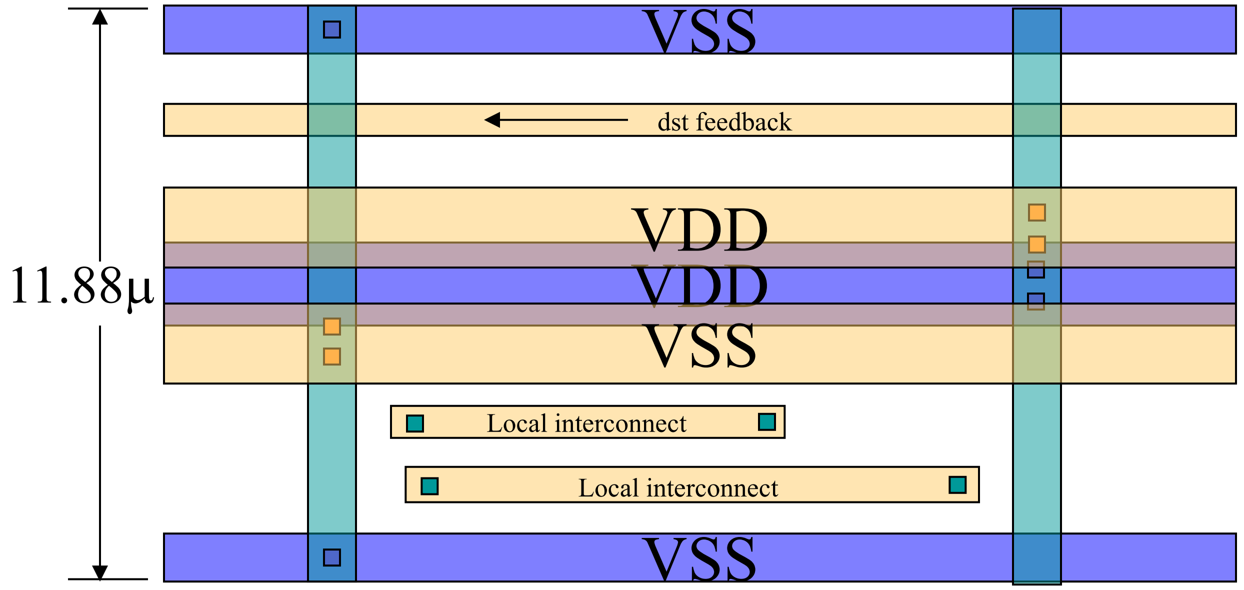
- **The bit-slice layout diagram given above is an example and would not need to be followed exactly. It is for reference as to how a bit-slice layout is normally done. 11.88m pitch however is required**
- **Metal direction conventions and power supply routing has to be done as shown below:**



Logic for 1-bit implementation

Other Details

- Power Gridding requirements:
  1) M1 lines run along bit slice at least 0.45u wide
  2) M2 powers intersect the M1 at least every 12u at 0.45u wide
  3) A M3 VSS and a M3 VDD of 0.99u have to deliver power to each slice of the bit slice.



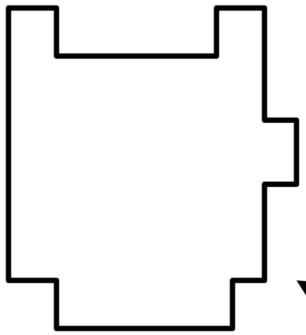
- M3 Usage
  1) *Dst* feedback which brings the *dst* bus back to your bypass mux
  2) Local interconnect within cells at your discretion

Evaluation Criteria

- Quality of report, schematics, & layout
- Normalized metric ty score, normalized to highest class score
- Metric = $\left(\frac{Frequency_{max}}{Power@50MHz\times Area}\right)^{m}$
- m is a factor (0.5 <= m <= 1.0) that can be used to control the spread of the grades. Basically, a knob we can tweak in case one group really does well and sets a standard other groups cannot meet
- The design must be able to run at 50MHz error free with a VDD of 1.8V.
- If this minimum frequency is not met the project will receive a failing grade. This minimum frequency is for the post-layout simulated netlist with all the parasitic capacitances

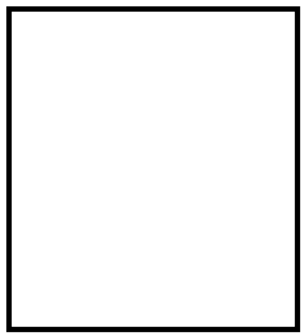
Evaluation Criteria (contd.)

- **Report does not need to contain lengthy text. Explain your adder architecture and why you chose it. Explain your motivation for flop choice (pulse vs M/S). Explain your thought process in sizing transistors.**
- **The following data is expected:**
  - – Max speed design could run post-layout (with ext. parasitics)
  - – Power consumption @50MHz post-layout.
  - – Area (rectilinear area)



Area to be reported is the area of a single rectangle that can encompass all the polygons of your design.

If your block looked like this, this is the area you would report.



Gate-level Design (submit by 8th Nov)

- **Blocks needed: adder, ~~multiplier~~, multiplexer, or any other you need**
  - ~~Multiplier needs shifter and adder~~
- **Slide 1: Datapath of the whole MLP**
  - You can re-use Slide 10 unless you make an improvement
- **Slide 2:** Gate-level implementations of your 4-bit adders
- **Slide 3:** Gate-level design of your shifter
- ~~**Slide 4:** Gate-level design of your multiplier~~
- **Slide 5:** Gate-level design of your multiplexer
- **Other slides:** Gate-level implementations of any other block you plan to use

Milestones

Remember the overall project weight is 25%

Milestone	Deadline	Weight in the project grade	Overall Weight
Team sign up*	Nov. 4th	0%	0%
Gate-level design (PDF/PPT)	Nov. 11th	20%	5%
Structural Verilog	Nov. 23rd	20%	5%
Unoptimized layout	Dec. 7th	20%	5%
Optimized layout	Dec. 14th	20%**	5%
Final project report	16th Dec	20%	5%

* You can work in teams of 3 or 4 (there are 61 students in total)

** Normalized across the class based on the quality metric