Verilog Implementation of Decision Tree Accelerator

Bo-Fan, Chen

Side Project for 2022 Job Interviews

Outline

- Problem/ Dataset
- Software
- Conversion
- Hardware

Problem Description

- Predict payment failure
    - Decision tree

- Decision tree can be used in various fields
    - Some of them require ***high processing speed***
        - One solution is to adopt a hardware accelerator dedicated for decision tree

Dataset

- Label
  - Payment success/fail, 0/1
- Features
  - 13 features for each application data
  - Including FICO score, interest rate…

- 9578 loaning data
  - 8045 successes / 1533 fails

Data Pre-process

- “Loaning Purpose” is a categorical feature
  - 6 categories
  - One hot encoding
  - 18 total features
- Normalize the features to 0~255, integer
  - **Decision thresholds will also be 0~255, integer!**
  - Hardware-friendly modification
  - At the cost of potential performance drop
- Training/Testing split
  - Randomly sample
  - 6704/2874

Outline

- Problem/ Dataset
- Software
- Conversion
- Hardware

Decision Tree

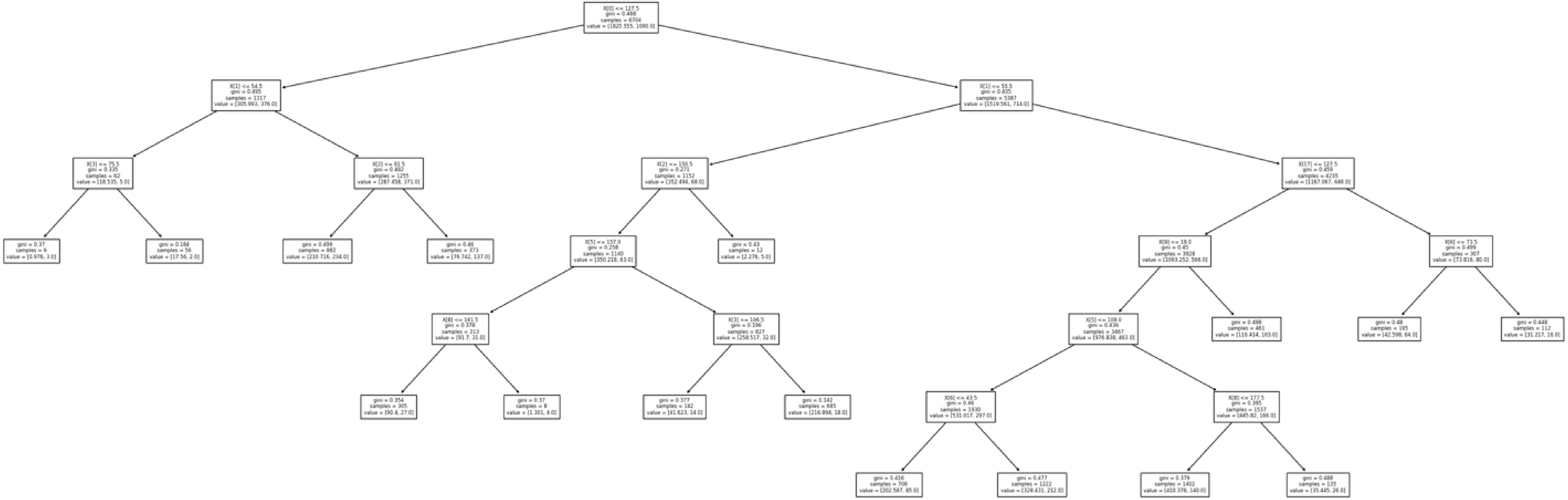
- Use the decision tree provided by scikit-learn
  - ***This work focused on the hardware implementation***
    - Fine-tuning and software analyses (exp: max-leaf-num, AUROC) are neglected
  - Confusion matrix on the test set (Before/ After converting to integers)

Ans\Pred	T	F
T	1970	461
F	284	159

Ans\Pred	T	F
T	1967	464
F	283	160

Result Tree Structure

- Max Node Index: 30
- Max-depth: 6



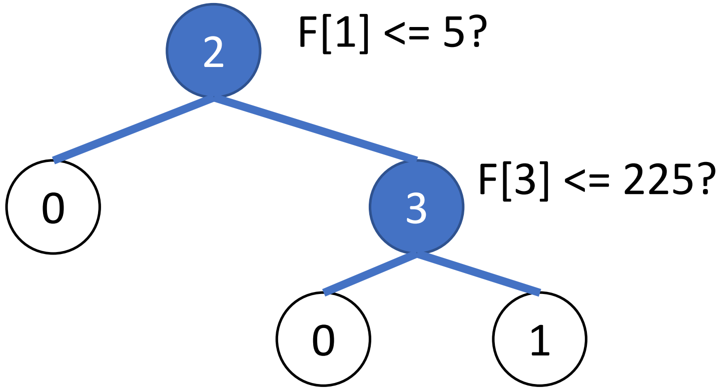
Outline

- Problem/ Dataset
- Software
- Conversion
- Hardware

Observation

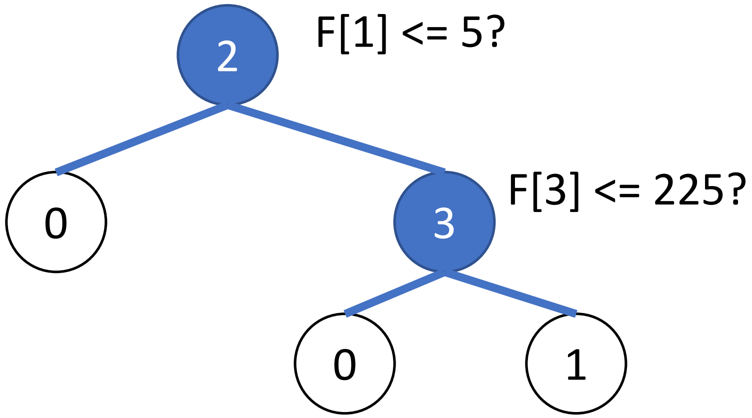
- Naïve thought
  - Core function is to “***move down 1 layer***”

- To move down 1 layer, we need 3 parameters:
  - The index of feature
  - The threshold
  - The index of the children



Conversion Strategy

- Convert python code to Verilog hardware design
  - Parse all the parameters into 3 memories
  - Leaf nodes share the same slot



	Feature Index	Threshold	Child
Node 0	0	0	0
	0	0	0
Node 2 (root node)	1	5	0
	3	225	0
	0	0	0
	0	0	3
	0	0	0
	0	0	1

Feature/Node Pruning

- The decision tree generated from scikit-learn can be further optimized

- Not all 18 features are used
  - Prune the unused ones. Re-index the features left.

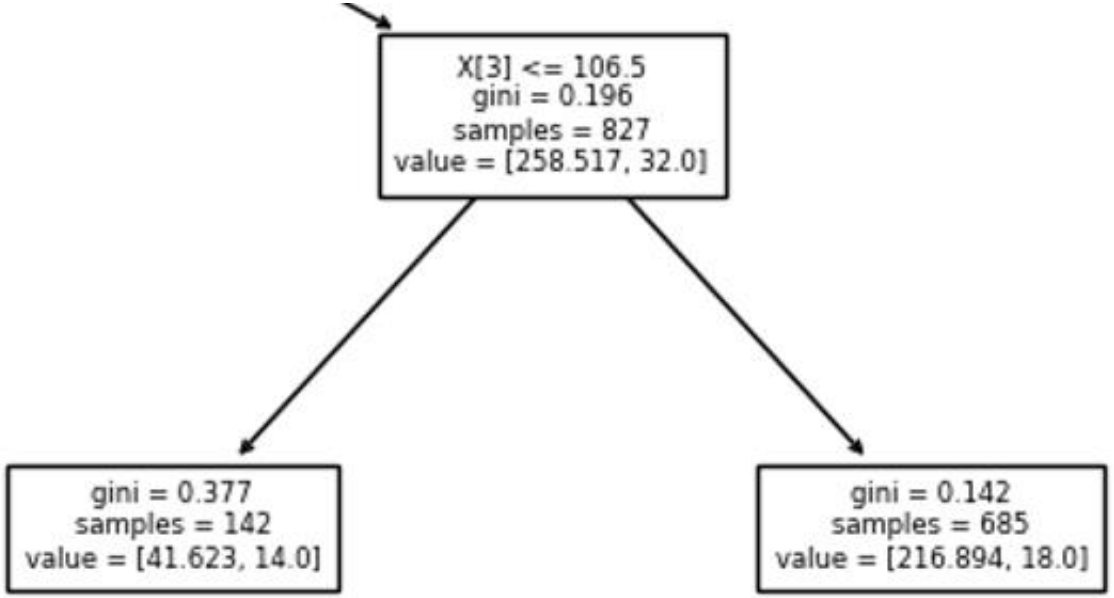
- Not all 31 node indexes are occupied
  - Leaf nodes do not need a unique index!
  - Prune the unused ones. Re-index the nodes.

Redundant Splits Pruning

- Reason: Decision tree training target
  - Minimizing the weighted entropy sum after split
    - ***Does not guarantee the split produces different predictions!***
  - Redundant splits are sometimes generated
  - Recursively prune the redundant splits

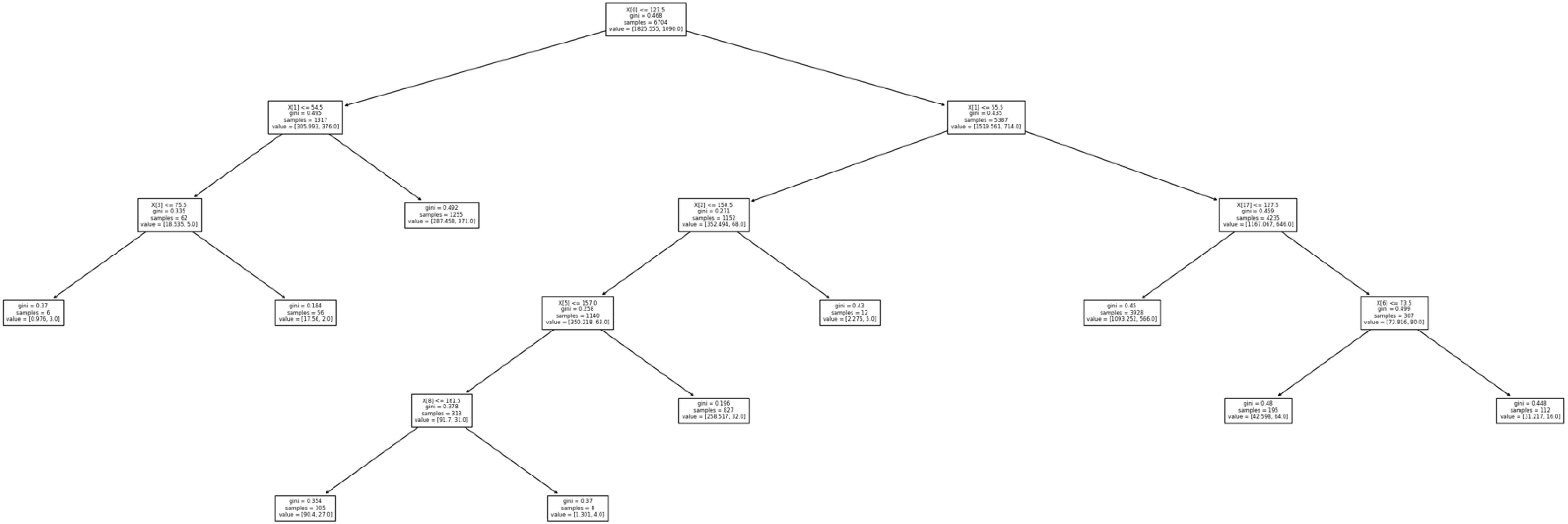
$0.377*142+0.142*685<0.196*827$

But both leaf nodes predict class 0!



Final Tree Structure

- Max node index: 10
- Max-depth: 5 layers

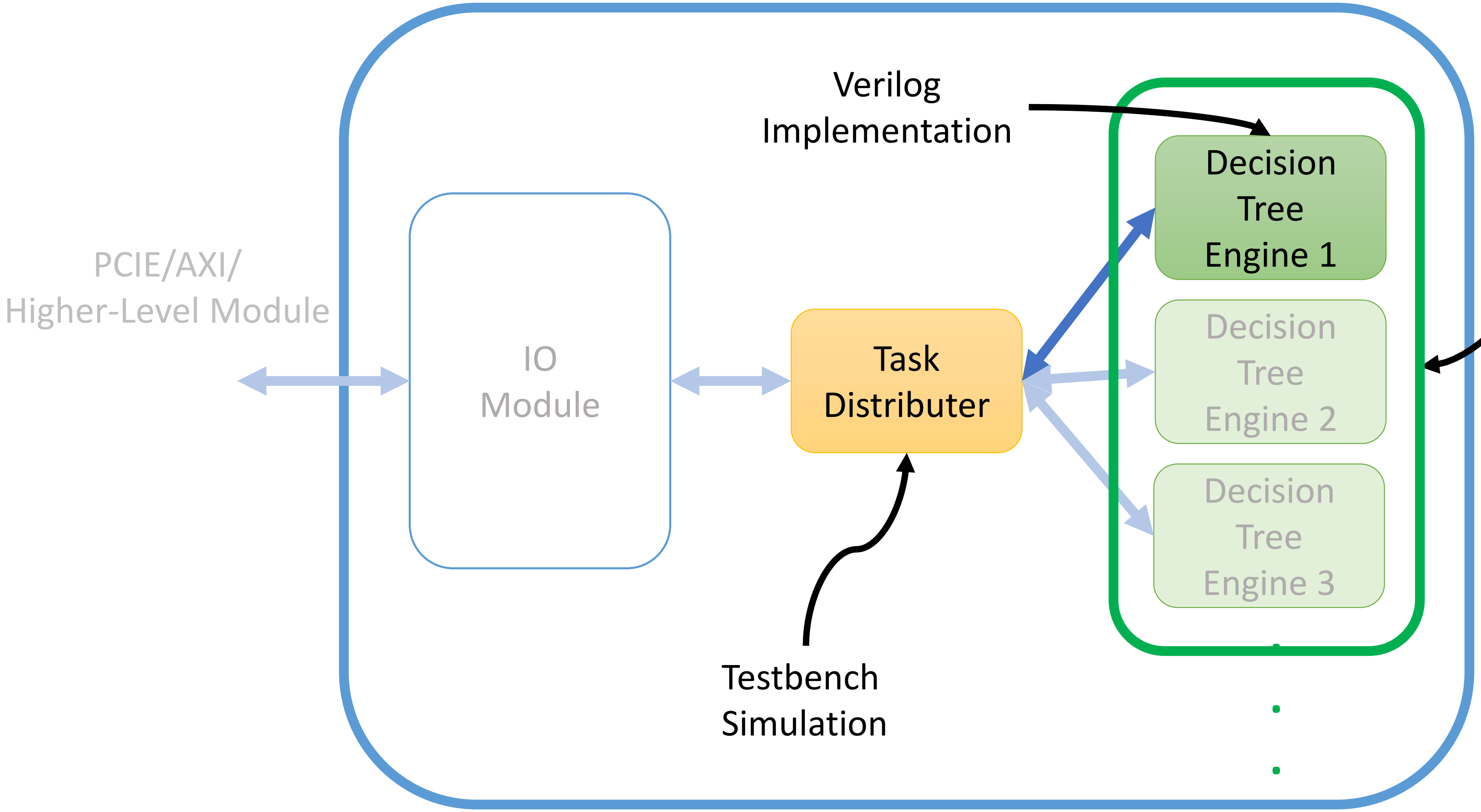


Outline

- Problem/ Dataset
- Software
- Conversion
- Hardware

Top-Level View / Project Scope

Can support Multiple Engines, but use 1 in this PoC work

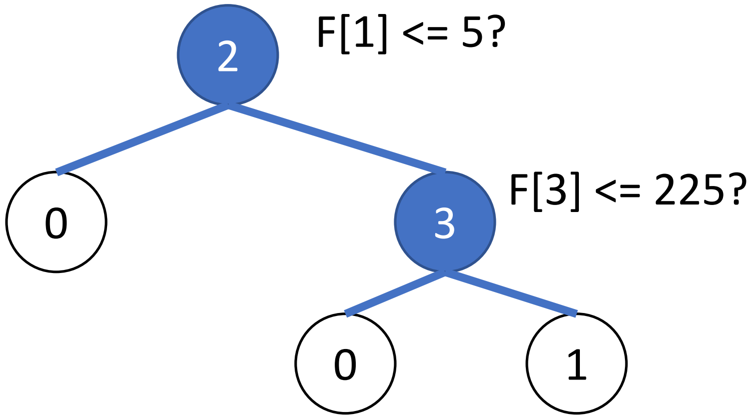


Architecture – Core Function

- 3-stage pipeline to move 1 layer down in the tree
  - 1st:
    - Retrieve the feature index by the node index
  - 2nd:
    - Select the feature using feature index
    - Retrieve the threshold used
  - 3rd:
    - Comparing the selected feature with the threshold
    - Retrieve index of the next node

Architecture – Core Function

- Parameter storage option
  - SRAM: Better generalizability for larger trees!
  - Flip-Flop array

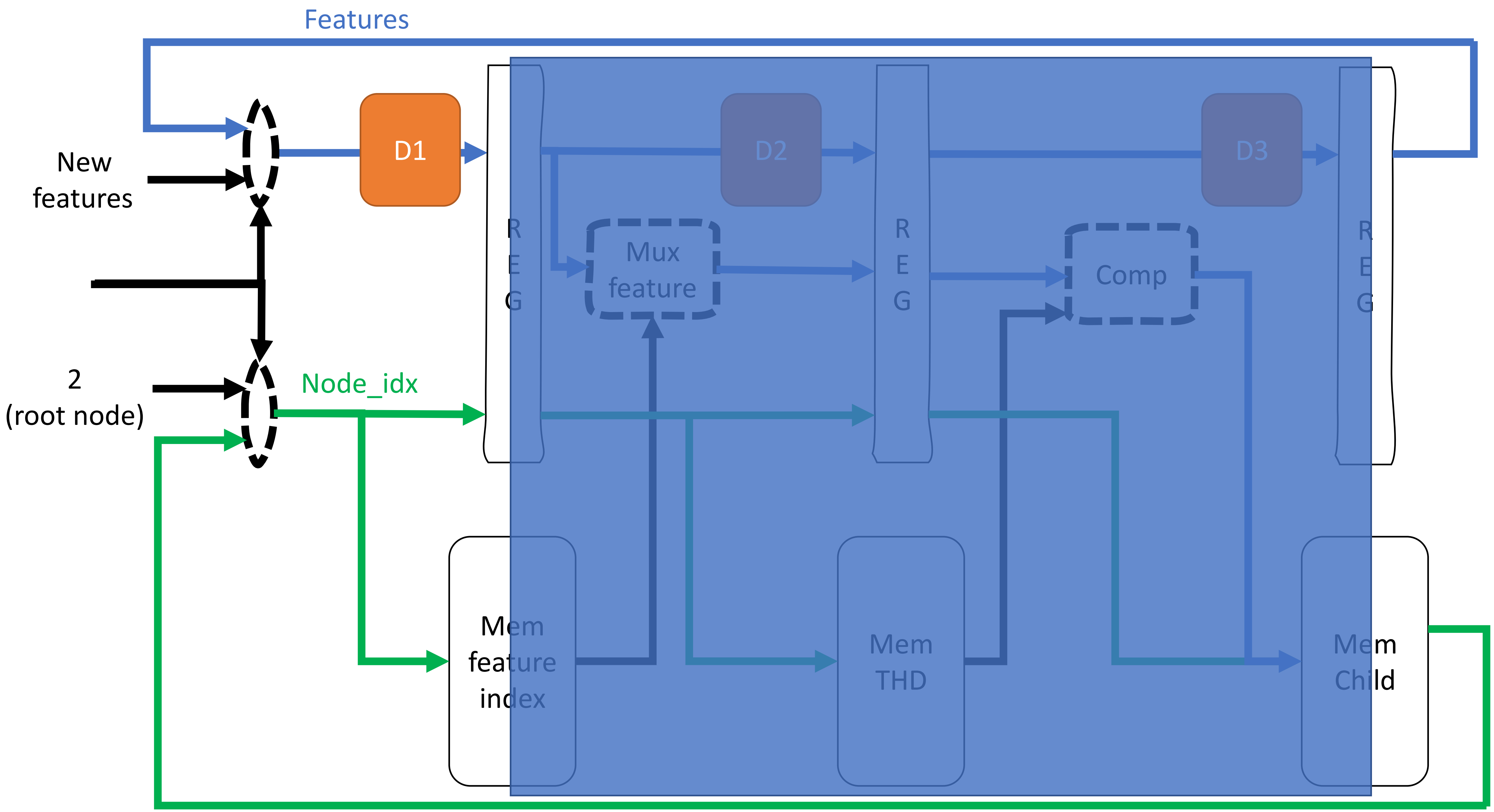


Node 0

Node 2 (root node)

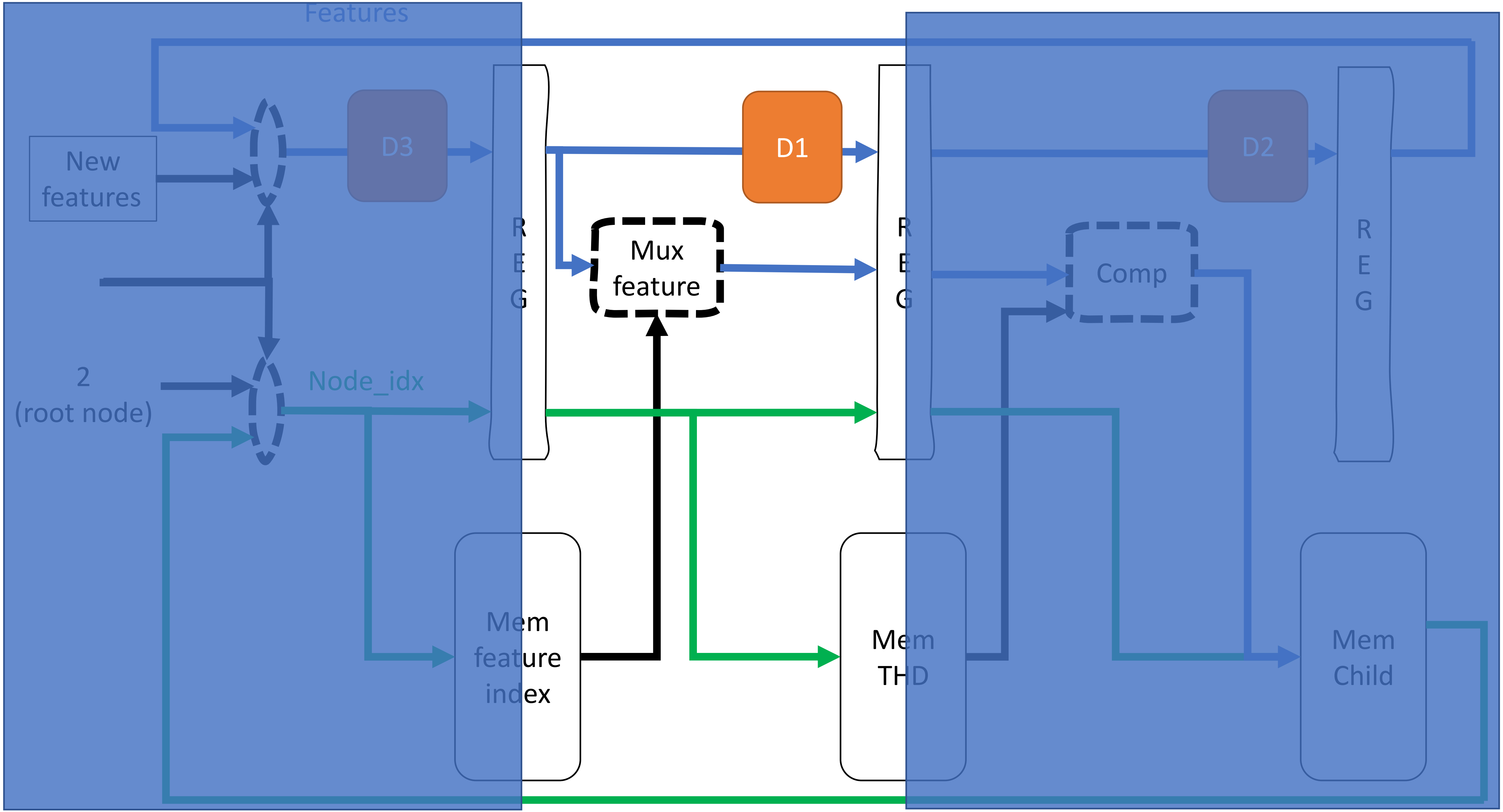
Feature Index	Threshold	Child
0	0	0
0	0	0
1	5	0
3	225	0
0	0	0
0	0	3
0	0	0
0	0	1

Stage 1



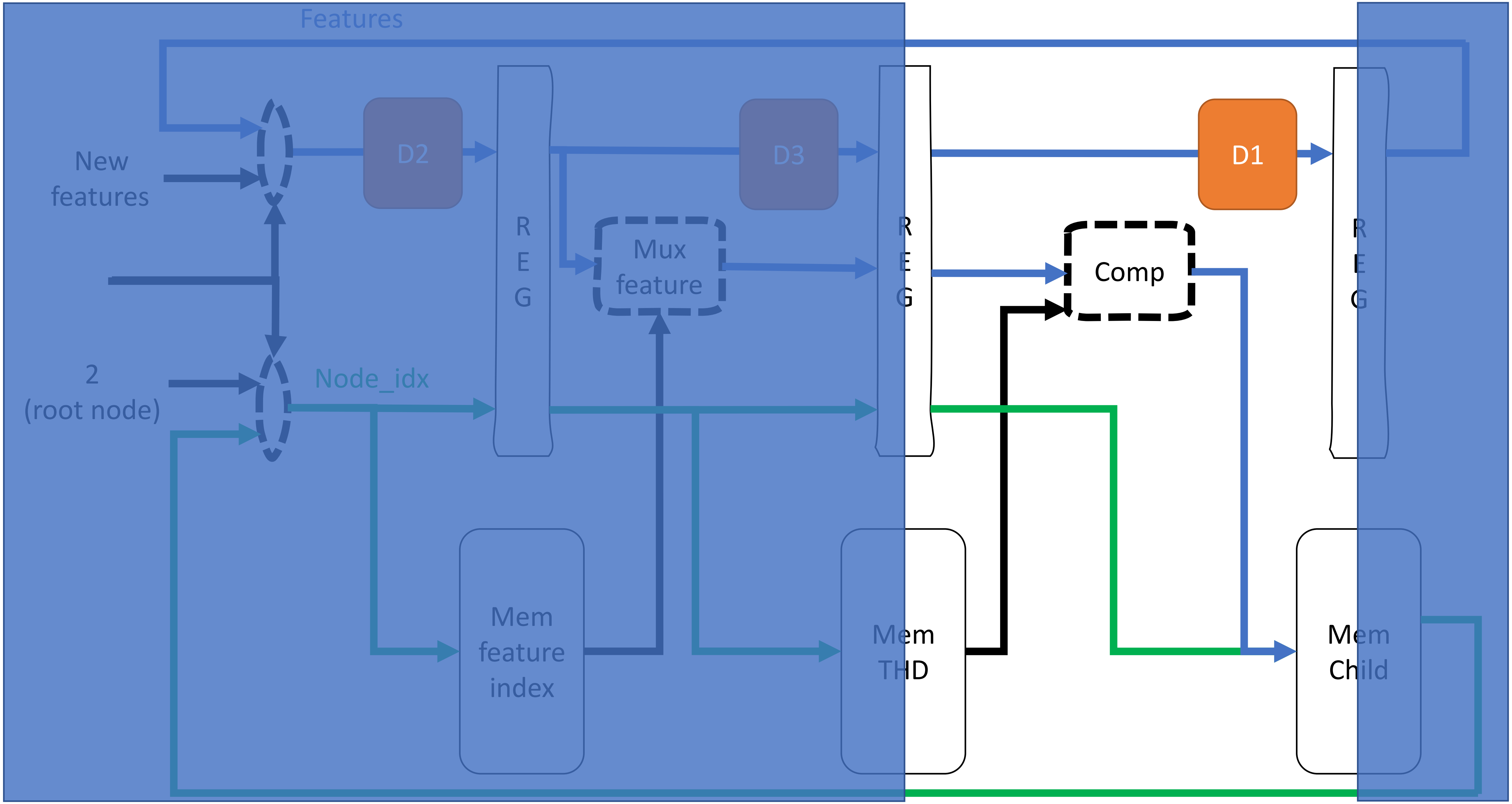
Features
New features
D1
R E G
D2
Mux feature
R E G
D3
Comp
R E G
2 (root node)
Node_idx
Mem feature index
Mem THD
Mem Child

Stage 2



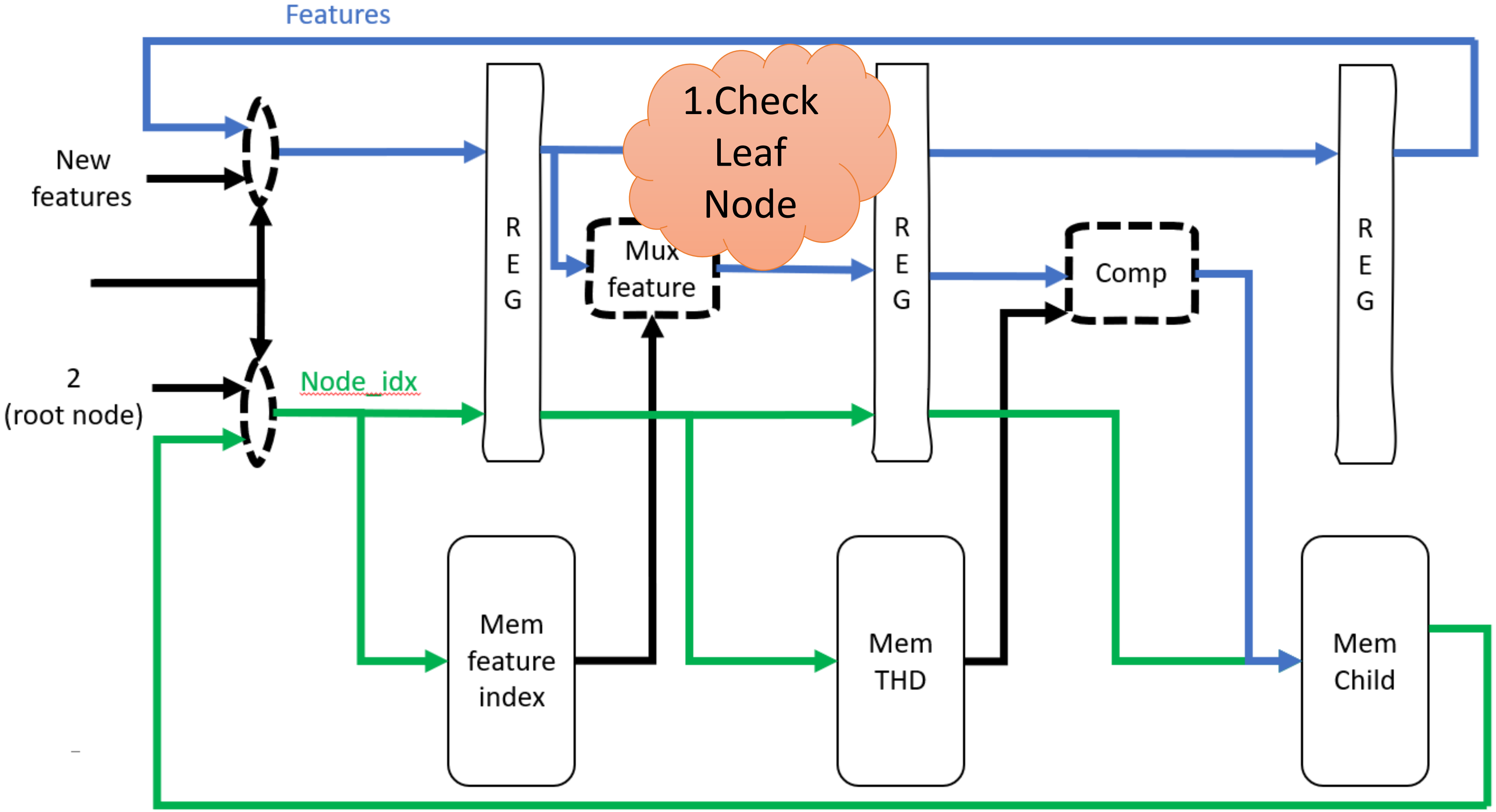
Features
New features
D3
R E G
D1
Mux feature
R E G
D2
Comp
R E G
2 (root node)
Node_idx
Mem feature index
Mem THD
Mem Child

Stage 3


Features
New features
D2
R E G
D3
R E G
D1
R E G
Mux feature
Comp
2 (root node)
Node_idx
Mem feature index
Mem THD
Mem Child

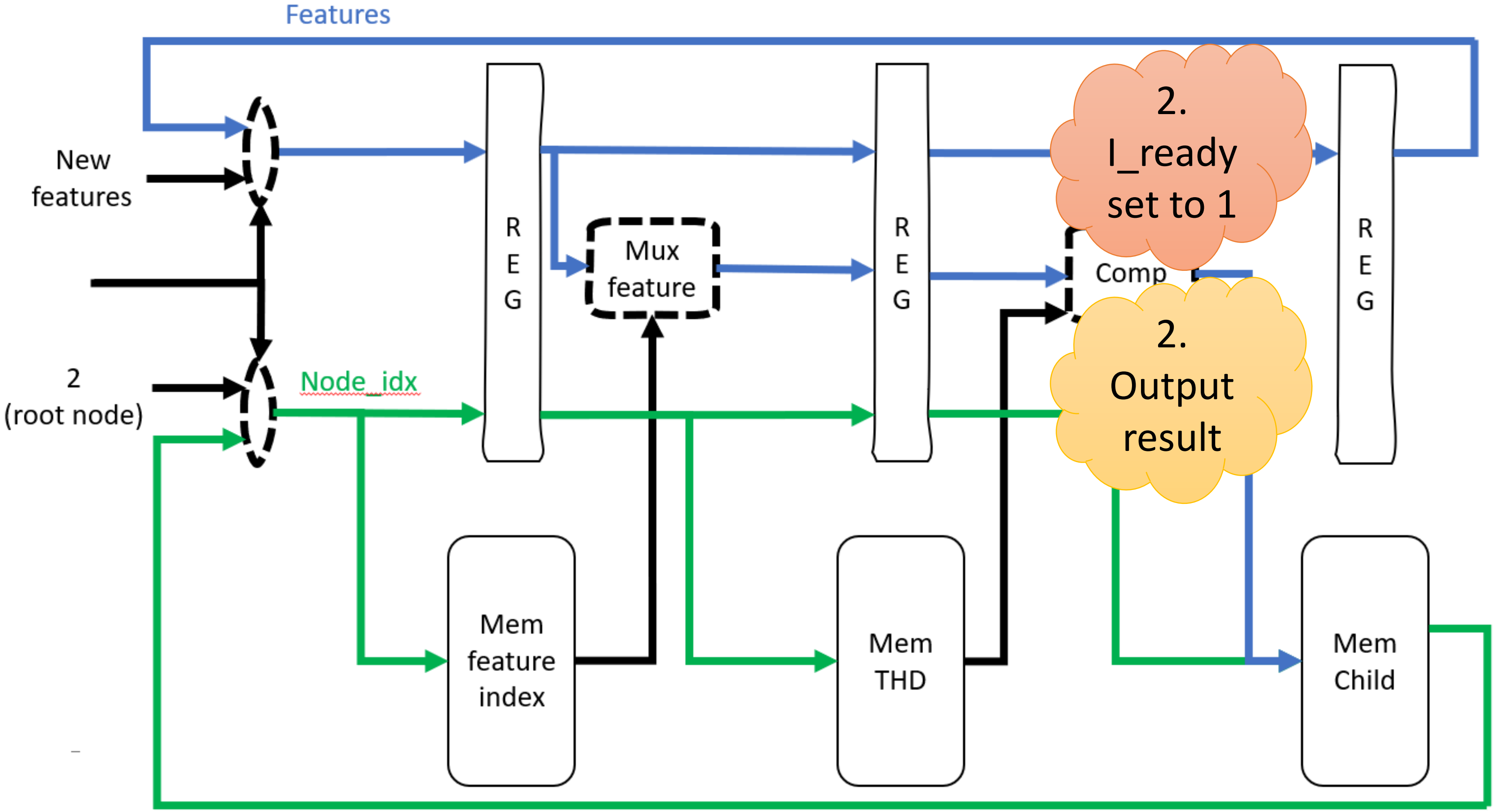
Architecture – A deeper look

- Input / Output
  - Assume the task distributer takes 1 cycle to respond



Architecture – A deeper look

- Input / Output
  - Assume the task distributer takes 1 cycle to respond

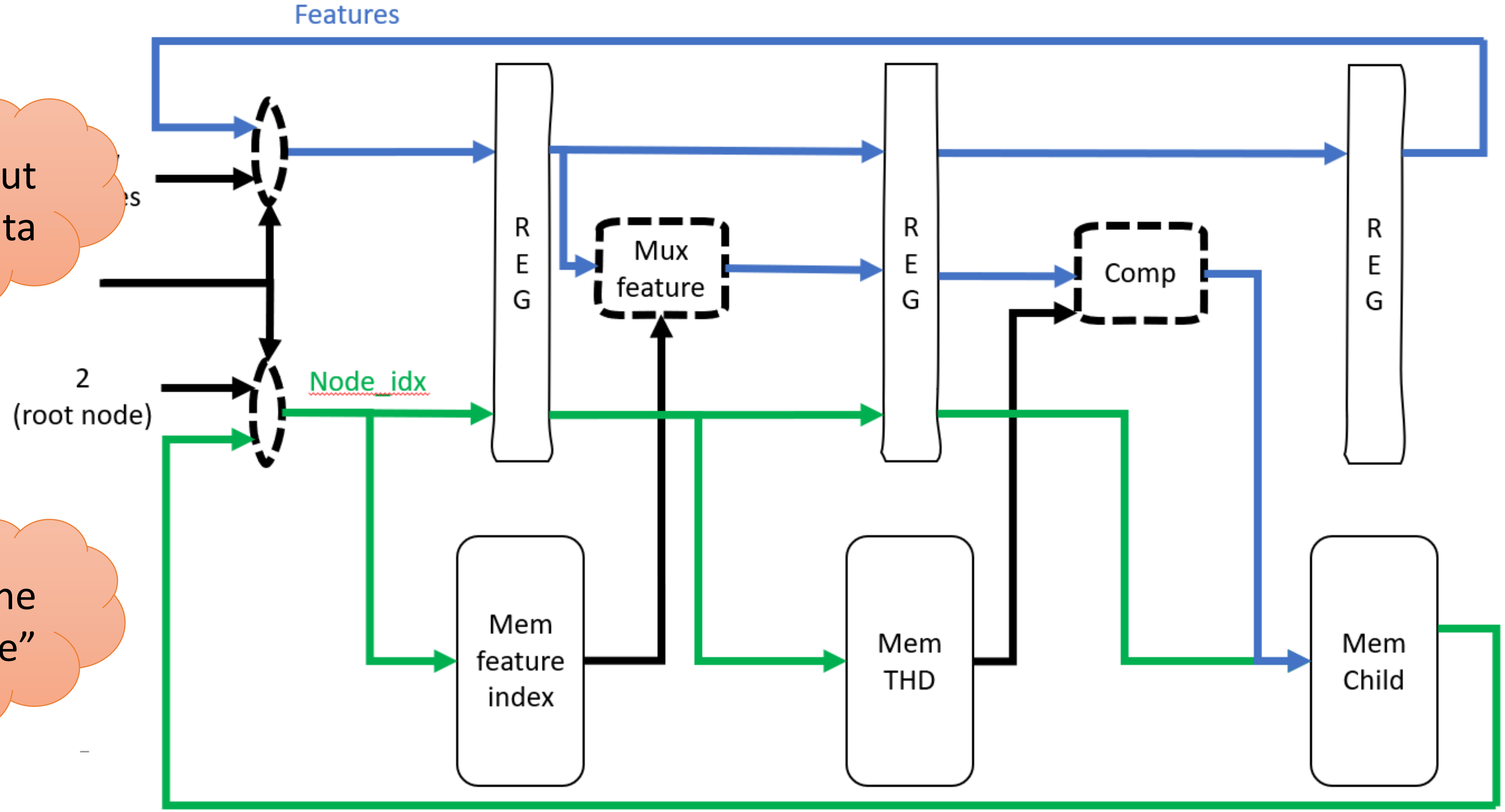


2. I_ready set to 1

2. Output result

Architecture – A deeper look

- Input / Output
  - Assume the task distributer takes 1 cycle to respond

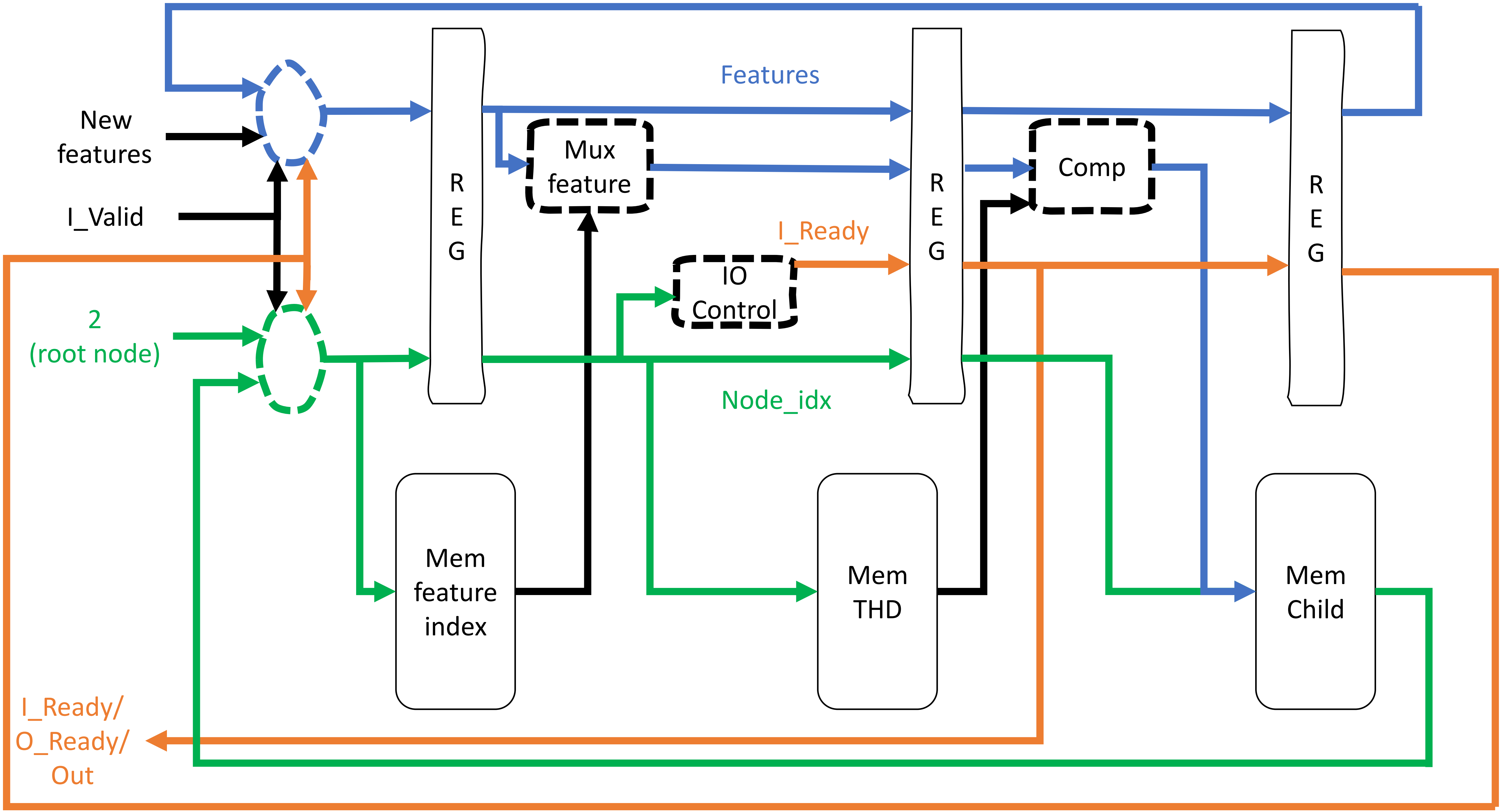


If new data is available

Input new data

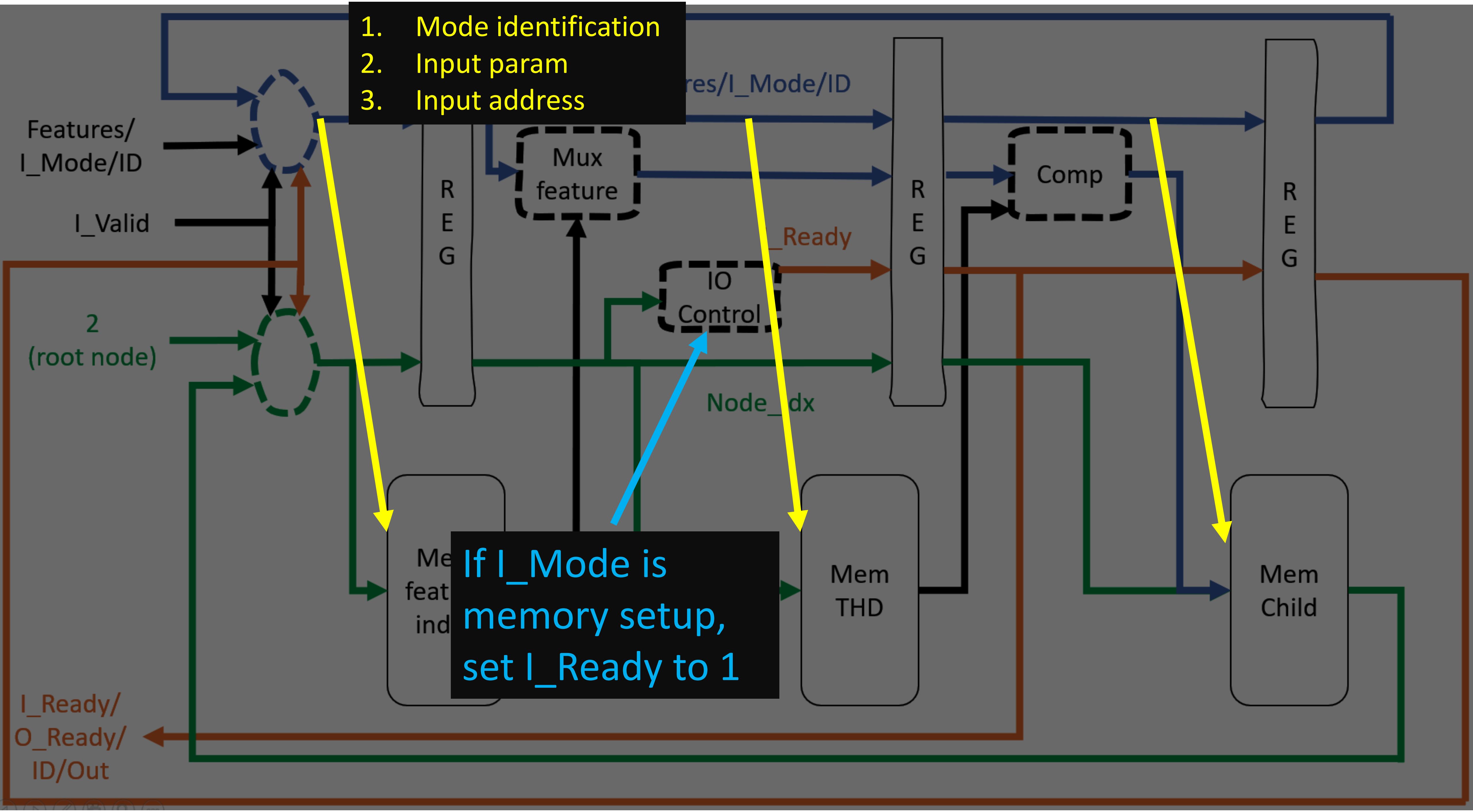
What if data is not valid?

Keep the “Bubble”


Features
New features
I_Valid
2 (root node)
R E G
Mux feature
R E G
Comp
R E G
I_Ready
IO Control
Node_idx
Mem feature index
Mem THD
Mem Child
I_Ready/ O_Ready/ Out

Architecture – A deeper look

- Memory Initialization
    - Pass the params and addresses one by one
        - Re-using existing input ports
    - How to identify the correct instruction?
        - Add a new input port, I_Mode
        - 00: Initialize feature index memory
        - 01: Initialize threshold memory
        - 10: Initialize child node memory
        - 11: Computation
    - Allows parameter modification at run time!

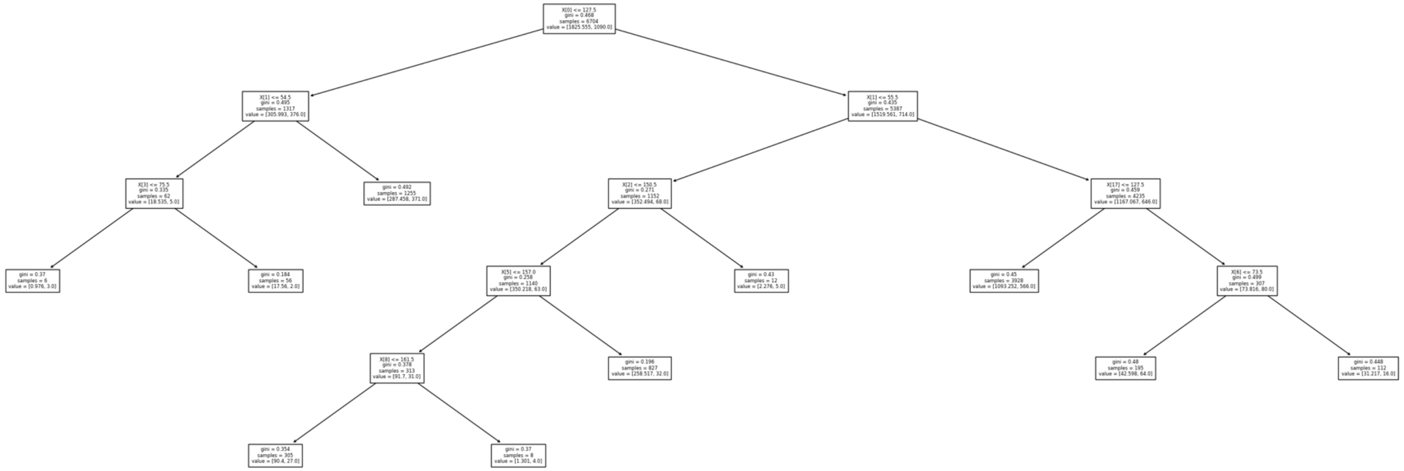


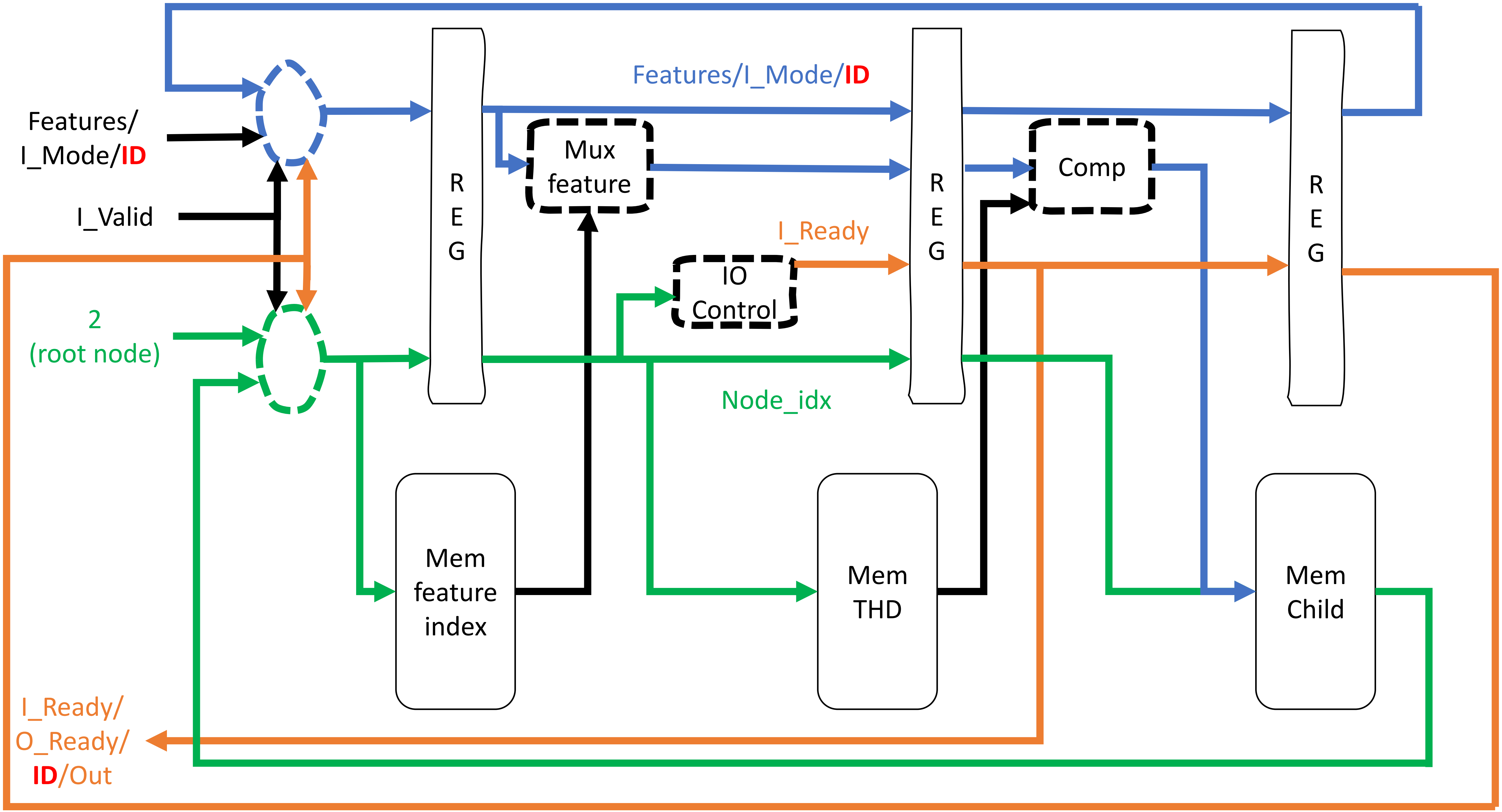
1. Mode identification
2. Input param
3. Input address

If I_Mode is memory setup, set I_Ready to 1

Architecture – Final thoughts

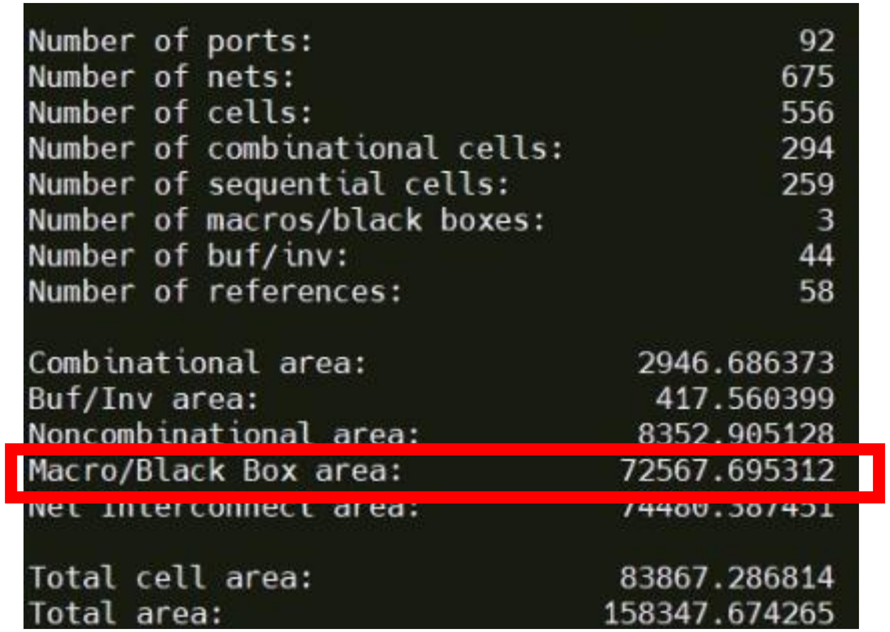
- Processing latency varies between data
    - IO does not follow first-in-first-out
    - Naïve solution:
        - Track the data ID in each stage


X[0] <= 127.5 gini = 0.468 samples = 6704 value = [1825.555, 1090.0]
X[1] <= 54.5 gini = 0.495 samples = 1317 value = [305.993, 376.0]
X[1] <= 55.5 gini = 0.435 samples = 5387 value = [1519.561, 714.0]
X[3] <= 75.5 gini = 0.335 samples = 62 value = [18.535, 5.0]
gini = 0.492 samples = 1255 value = [287.458, 371.0]
X[2] <= 150.5 gini = 0.271 samples = 1152 value = [352.494, 68.0]
X[17] <= 127.5 gini = 0.459 samples = 4235 value = [1167.067, 646.0]
gini = 0.37 samples = 6 value = [0.976, 3.0]
gini = 0.184 samples = 56 value = [17.56, 2.0]
X[5] <= 157.0 gini = 0.258 samples = 1140 value = [350.218, 63.0]
gini = 0.43 samples = 12 value = [2.276, 5.0]
gini = 0.45 samples = 3928 value = [1093.252, 566.0]
X[6] <= 73.5 gini = 0.499 samples = 307 value = [73.816, 80.0]
X[8] <= 161.5 gini = 0.378 samples = 313 value = [91.7, 31.0]
gini = 0.196 samples = 827 value = [258.517, 32.0]
gini = 0.48 samples = 195 value = [42.598, 64.0]
gini = 0.448 samples = 112 value = [31.217, 16.0]
gini = 0.354 samples = 305 value = [90.4, 27.0]
gini = 0.37 samples = 8 value = [1.301, 4.0]

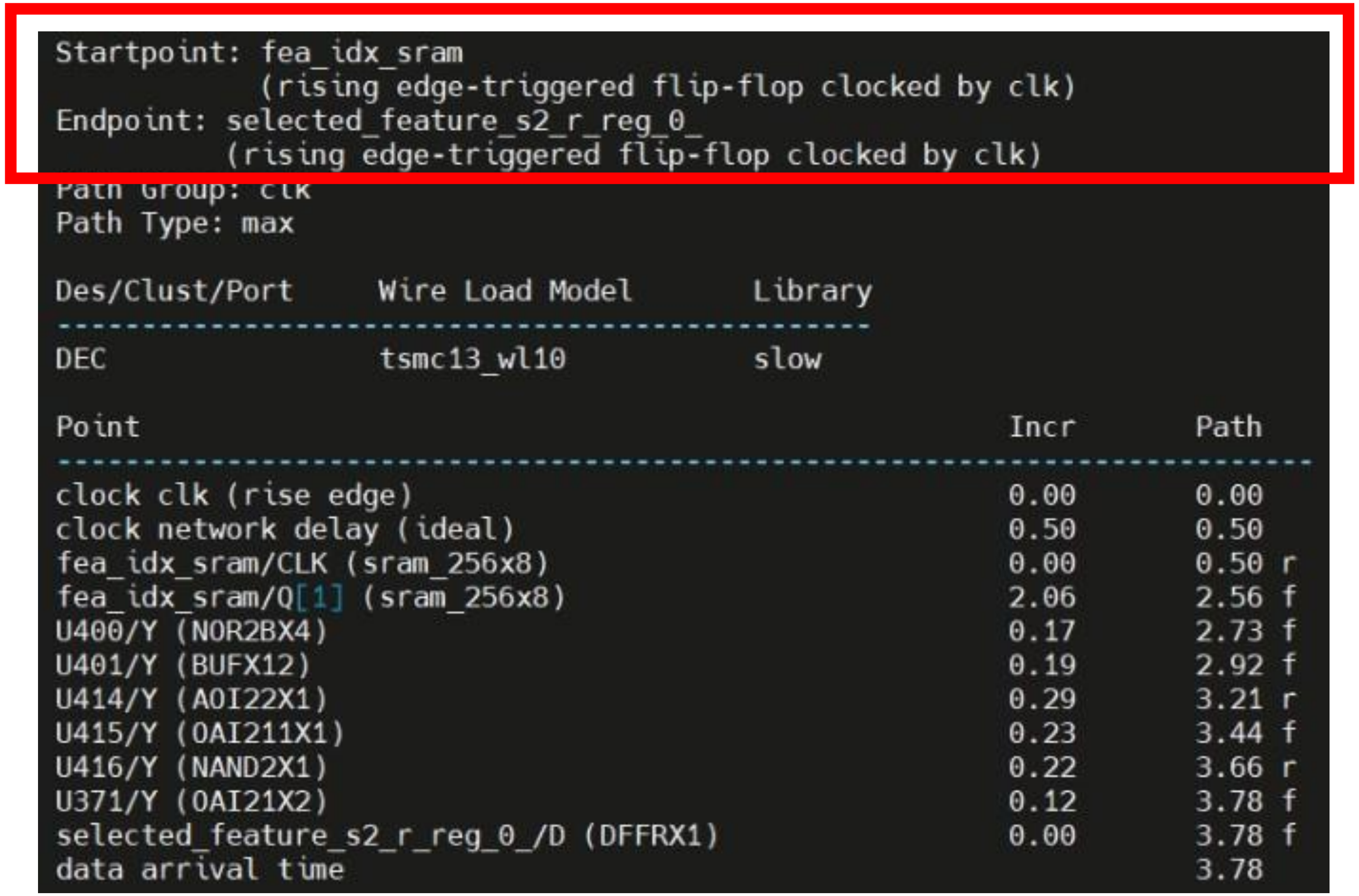

Features/
I_Mode/ID
I_Valid
2
(root node)
R
E
G
Features/I_Mode/ID
Mux
feature
IO
Control
I_Ready
Node_idx
R
E
G
Comp
R
E
G
Mem
feature
index
Mem
THD
Mem
Child
I_Ready/
O_Ready/
ID/Out

Synthesis

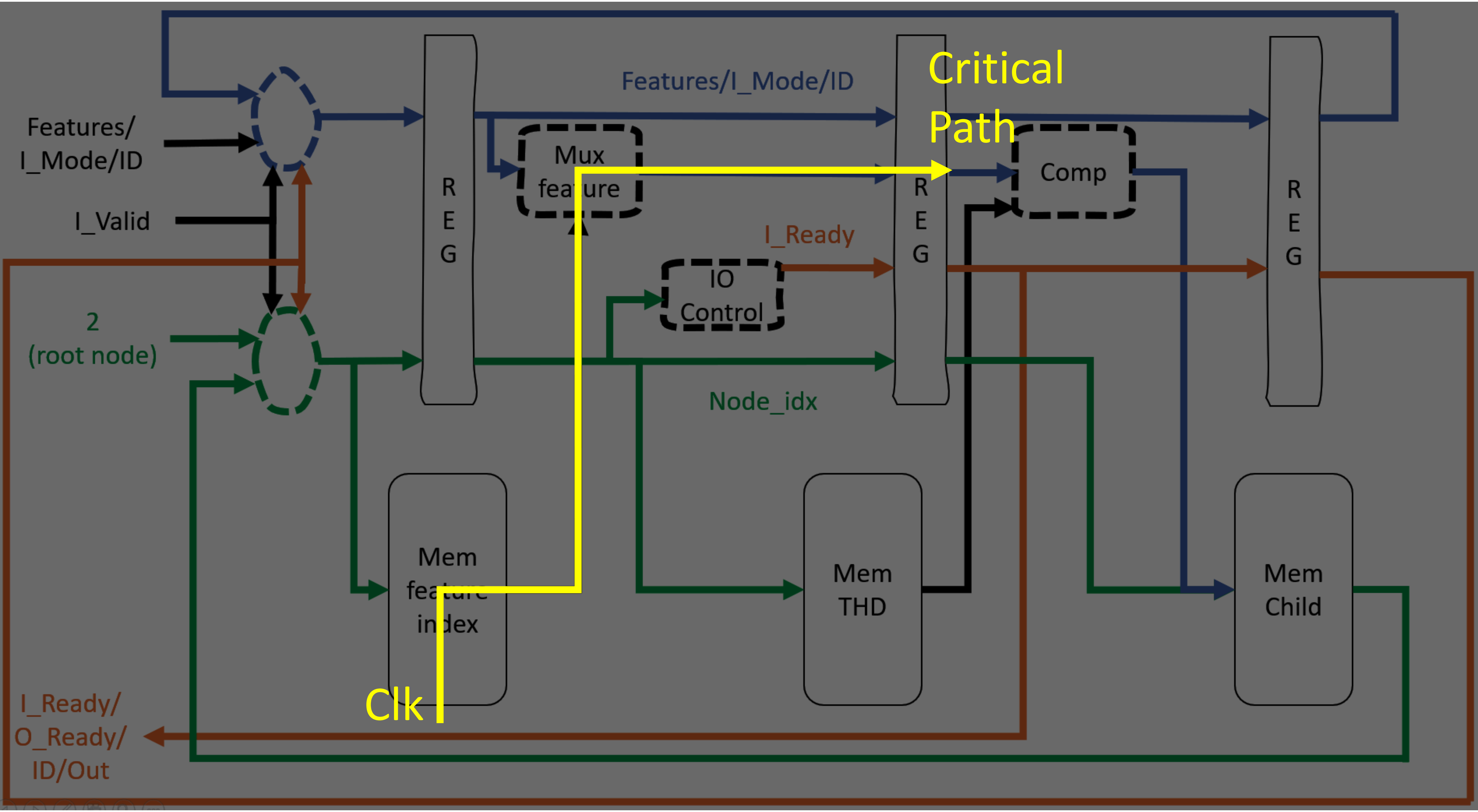
- Using Synopsys Design Vision
  - Process node: TSMC 0.13um
  - Clock Cycle: 3.6ns
  - Memory: 86% of the cell area



```
Number of ports:                               92
Number of nets:                               675
Number of cells:                              556
Number of combinational cells:                294
Number of sequential cells:                   259
Number of macros/black boxes:                   3
Number of buf/inv:                             44
Number of references:                          58

Combinational area:                   2946.686373
Buf/Inv area:                          417.560399
Noncombinational area:                8352.905128
Macro/Black Box area:                72567.695312
Net Interconnect area:               74480.387451

Total cell area:                     83867.286814
Total area:                         158347.674265
```

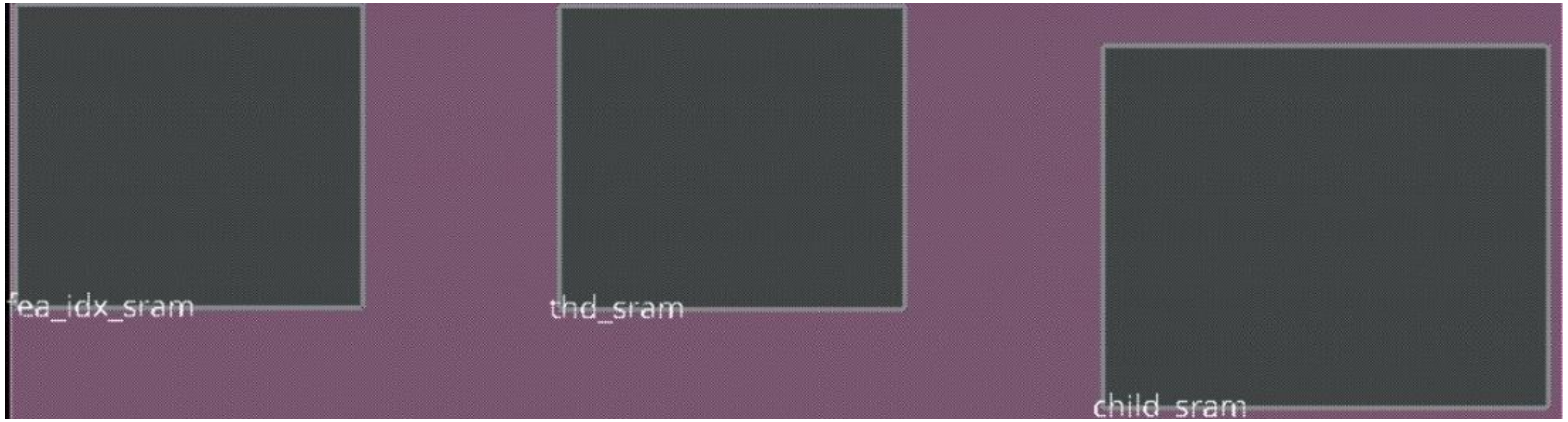


```
Startpoint: fea_idx_sram
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: selected_feature_s2_r_reg_0_
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Des/Clust/Port     Wire Load Model       Library
------------------------------------------------
DEC                tsmc13_wl10           slow

Point                                                  Incr       Path
-------------------------------------------------------------------------
clock clk (rise edge)                                  0.00       0.00
clock network delay (ideal)                            0.50       0.50
fea_idx_sram/CLK (sram_256x8)                          0.00       0.50 r
fea_idx_sram/Q[1] (sram_256x8)                         2.06       2.56 f
U400/Y (NOR2BX4)                                       0.17       2.73 f
U401/Y (BUFX12)                                        0.19       2.92 f
U414/Y (AOI22X1)                                       0.29       3.21 r
U415/Y (OAI211X1)                                      0.23       3.44 f
U416/Y (NAND2X1)                                       0.22       3.66 r
U371/Y (OAI21X2)                                       0.12       3.78 f
selected_feature_s2_r_reg_0_/D (DFFRX1)                0.00       3.78 f
data arrival time                                                 3.78
```

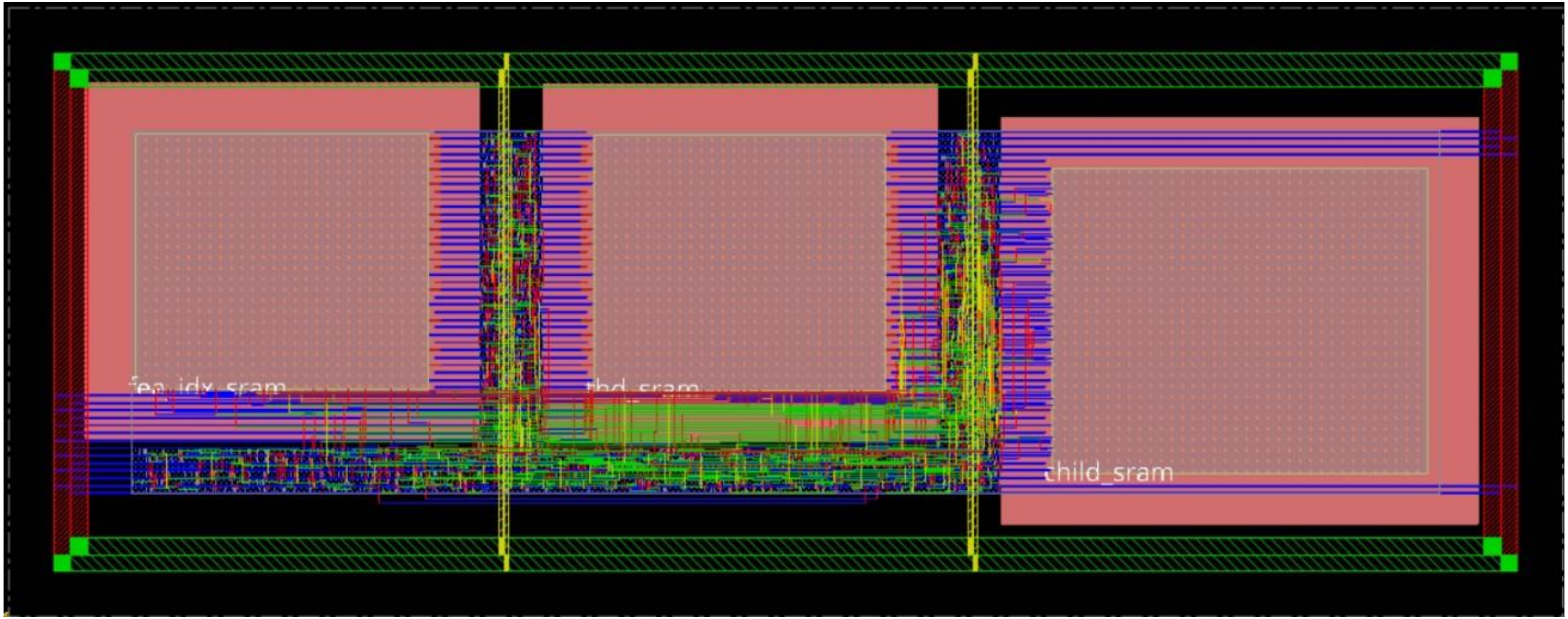

Critical
Path
Features/I_Mode/ID
Features/
I_Mode/ID
I_Valid
2
(root node)
R
E
G
Mux
feature
I_Ready
IO
Control
Comp
Node_idx
Mem
feature
index
Mem
THD
Mem
Child
Clk
I_Ready/
O_Ready/
ID/Out

Place and Route

- Using Cadence Innovus
  - Utilization Rate: 0.65
  - Memory dominating layout
    - **Floorplan, memory placement, power plan** are critical to avoid violations!
  - Pass P&R testbench simulation at clock cycle 3.6 ns

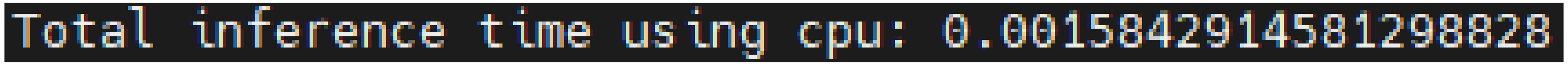


Place and Route



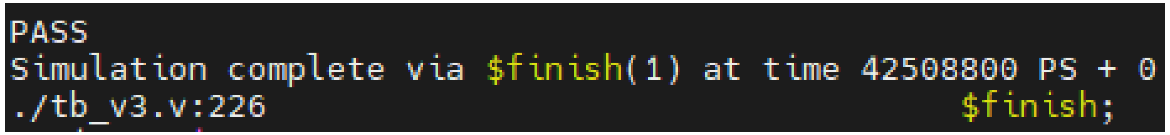
Accelerator Performance

- The evaluation metric is the inference time of all 2874 testing data.
- The baseline inference time of scikit-learn decision tree, using CPU, is 1.58 ms
  - Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz



```
Total inference time using cpu: 0.0015842914581298828
```

- The presented decision tree accelerator only requires 42.5 us!
  - Including memory setup
  - ~37 times faster!



```
PASS
Simulation complete via $finish(1) at time 42508800 PS + 0
./tb_v3.v:226                                    $finish;
```

References

1. https://github.com/Shayan-Asgari/ClassificationTrees
2. https://stackoverflow.com/questions/51397109/prune-unnecessary-leaves-in-sklearn-decisiontreeclassifier
3. https://stackoverflow.com/questions/56334210/how-to-extract-sklearn-decision-tree-rules-to-pandas-boolean-conditions
4. Course material of COMPUTER-AIDED VLSI SYSTEM DESIGN, National Taiwan University

About me

- Bo-Fan, Chen
- Nationality: Taiwan
- Language
  - Mandarin
  - English
- Education
  - National Taiwan University
    - Electric Engineering, graduated in 2019.07
    - Graduate Institute of Electronics Engineering, Integrated Circuits & Systems Group, Laboratory for Data Processing Systems, expect to graduate in 2022.09
  - Aarhus University, Denmark
    - Exchange student, from 2021.09 to 2022.07
    - Software Engineering, Electric Engineering
- Academic Work
  - CF-NET: Complementary Fusion Network for Rotation Invariant Point Cloud Completion, ICASSP 2022
- Skills
  - Digital Circuit Design, Verilog, Synopsys Design Vision, Cadence Innovus
  - Machine Learning, Python, Numpy, Pytorch, Tensorflow
- Objective
  - Looking for a position as digital circuit RTL engineer. Preferred locations are: United States, Europe, Taipei.

Contact

- Email
  - sddslover@gmail.com
- LinkedIn
  - linkedin.com/in/bo-fan-chen