fwXmachina

Nanosecond machine learning with boosted decision trees for high energy physics

Tae Min Hong

- *Paper* [2104.03408]
- *Info* fwx.pitt.edu
- *Code* gitlab.com/PittHongGroup/fwX

Pheno 2021
May 24, 2021
https://indico.cern.ch/event/982783/sessions/396894/

Thank you to my collaborators

[2104.03408]

fwx.pitt.edu

gitlab.com/PittHongGroup/fwX

PITT-PACC-2103-v2

Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics

T.M. Hong,* B.T. Carlson, B.R. Eubanks, S.T. Racz,† S.T. Roche,
J. Stelzer, and D.C. Stumpp‡

Department of Physics and Astronomy
University of Pittsburgh

May 17, 2021

Undergraduate researchers

† Received his BS last month!
‡ Now a Pitt PhD student in EE

Outline

- Intro
- Algorithm structure
- Firmware design
- Physics results (simulated)

- Backup slides

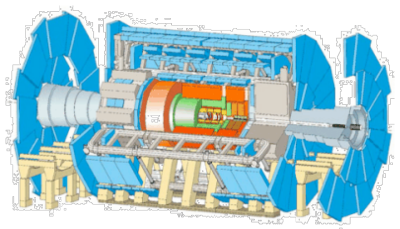
Outline (2)

- Intro
  - Machine learning at Level-1 trigger
- Algorithm structure
  - Bit integer representation
  - Tree flattening & merging
- Firmware design
  - Bin Engines
- Physics results (simulated)
  - VBF Higgs vs. multijet

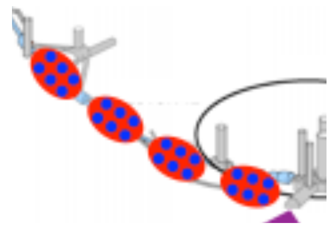
- Backup slides
  - Comparison to hls4ml
  - Test bench

Machine learning at L1 trigger

Event size · *Collision rate* = 60 TB/s



1.5 MB



40 MHz

Partial data

...
99
...
0

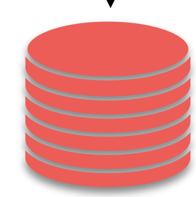
Full data buffer

Trigger system

- Processes partial data
- Decides what to save
- Not much time to decide

yes / no

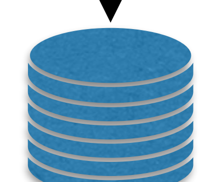
160 GB/s



Temporary storage

yes / no

1.5 GB/s



Permanent storage

Source: http://cern.ch/twiki/pub/Atlas/TDAQSpeakersCommitteeCommonReferences/tdaqFullNew2017.pdf

Machine learning at L1 trigger (2)

FW X Machina **This talk**

65 TB/s

Partial data

...
99
...
0

Level	*Latency*
L1	**O(1) µs**
HLT	O(1) s

FPGA

Custom electronics

Reduce rate to
100 kHz

yes / no

160 GB/s

50k CPUs

Software

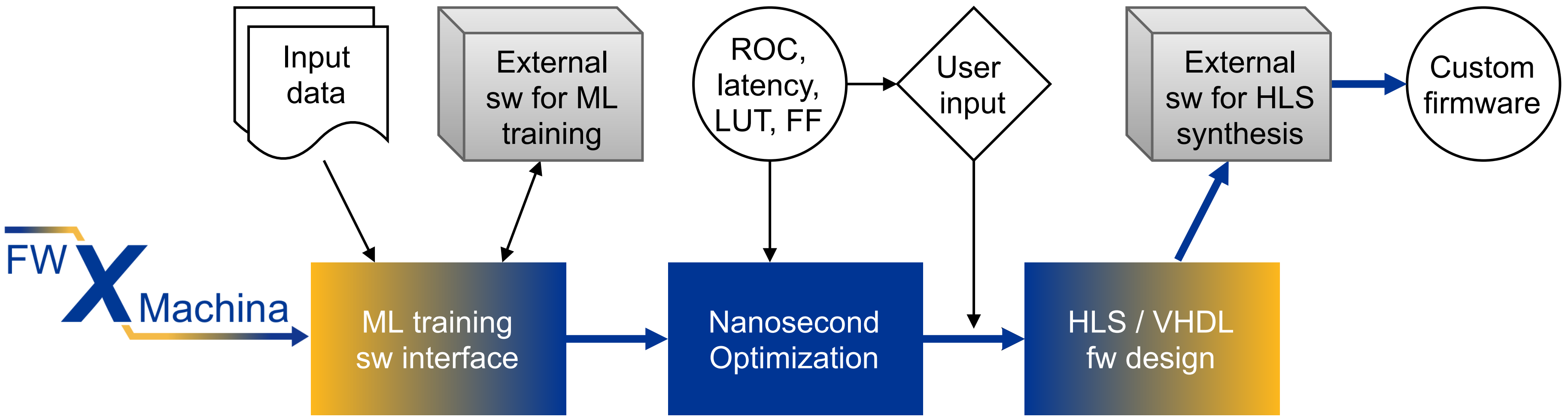
Reduce rate to
1-10 kHz

yes / no

1.5 GB/s

Source: http://cern.ch/twiki/pub/Atlas/TDAQSpeakersCommitteeCommonReferences/tdaqFullNew2017.pdf

- **Workflow**



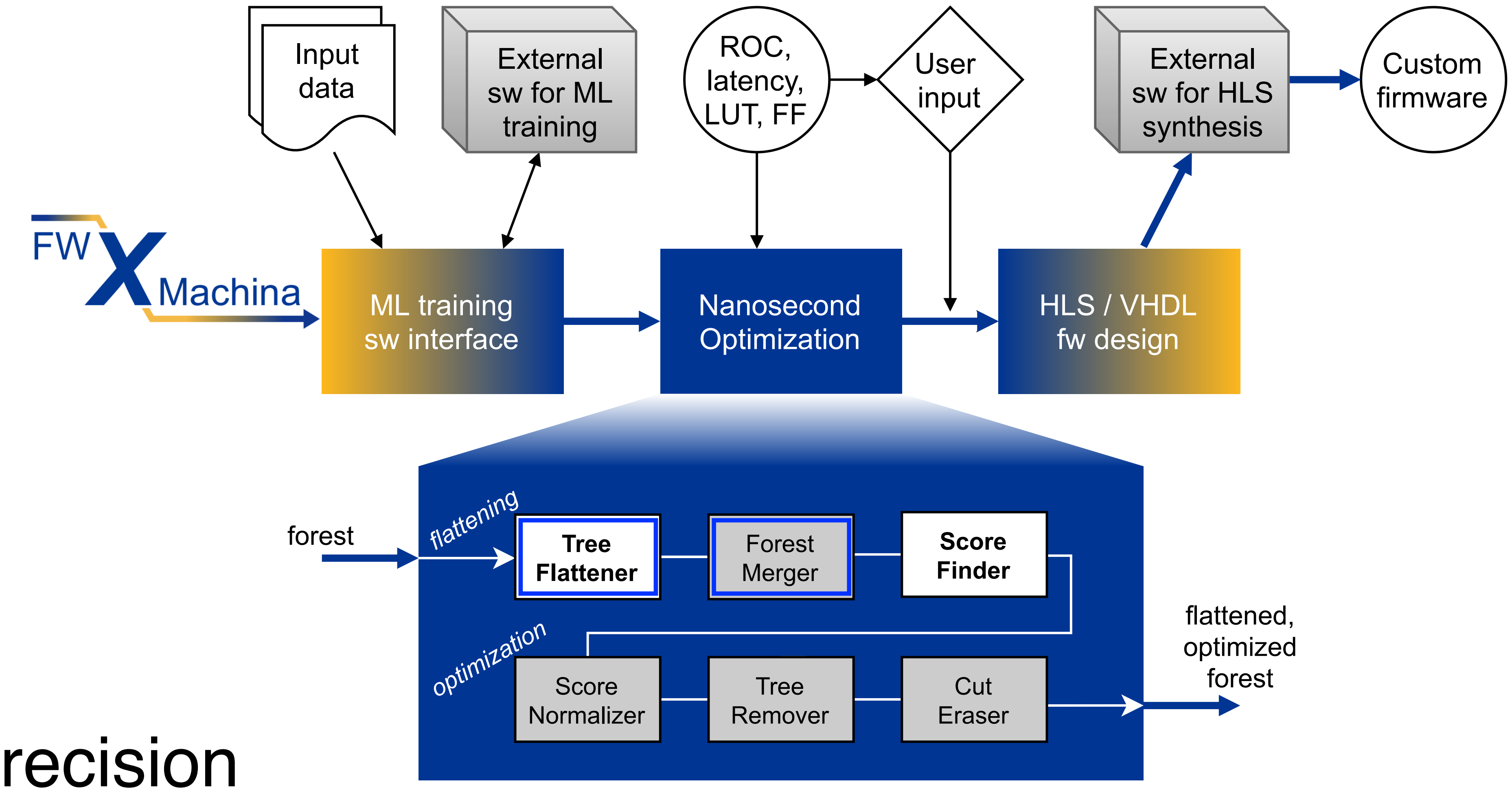
- Optimization

- Use bit integer precision

structure (2)

TM Hong

- Workflow
- **Optimization**
- Use bit integer precision



- Will discuss next:

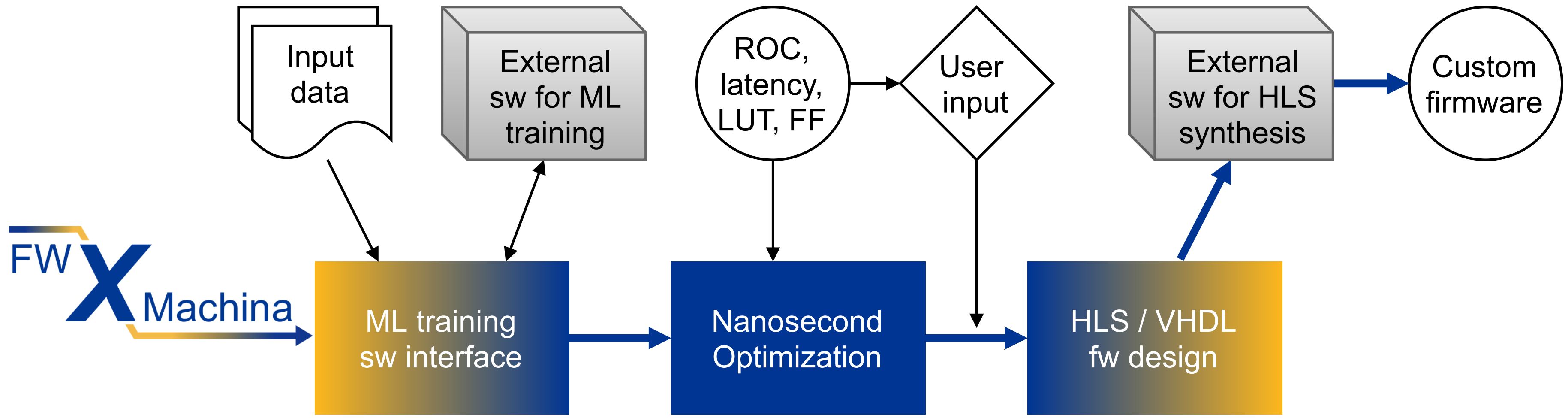
  Tree Flattener

  Forest Merger

FWXMachina structure (3)

TM Hong

- Workflow



- Optimization

- **Use bit integer precision**

  E.g., ap_int⟨8⟩ means the variable is represented by a range from 0 to 255.

- Advantages & subtleties

  Bit integers represents a wide range without sacrificing float precision

Transformation $c_{\text{int}} = f(c_{\text{float}}) = \left\lfloor \frac{c_{\text{float}} - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \cdot \left(2^N - 1\right) \right\rfloor$

Floor operation

Pre-evaluate f ***Firmware only adds***

$$f(x_1 + x_2) = f(x_1) + f(x_2)$$

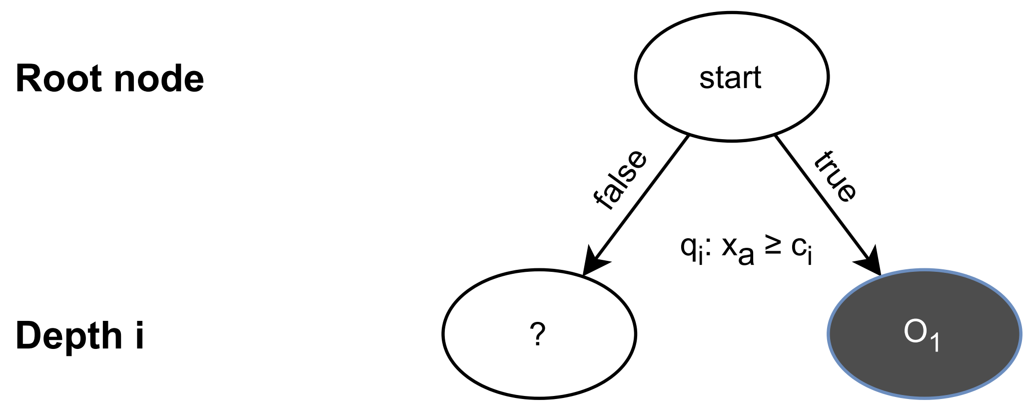
Equal up to one bit because of floor

Decision tree, 2 var example

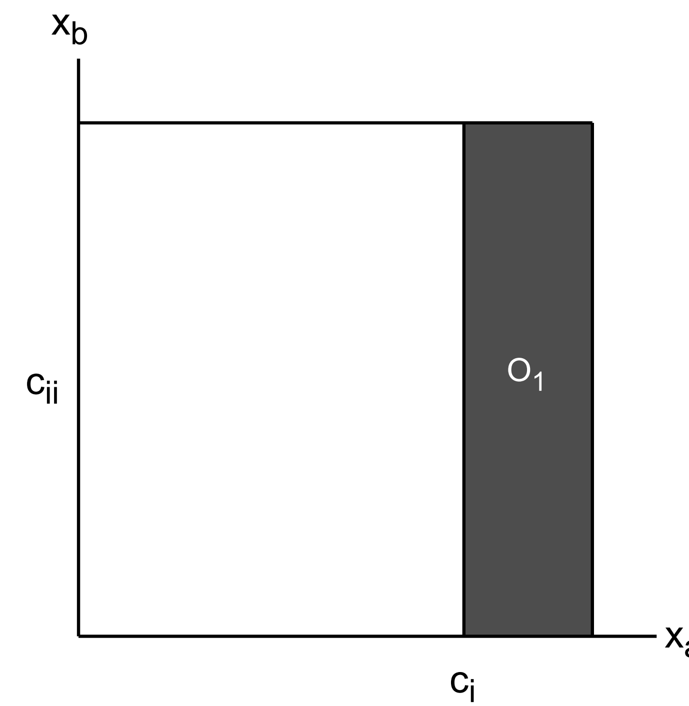
TM Hong

First step

Conventional tree structure



2d plane: x_a vs. x_b



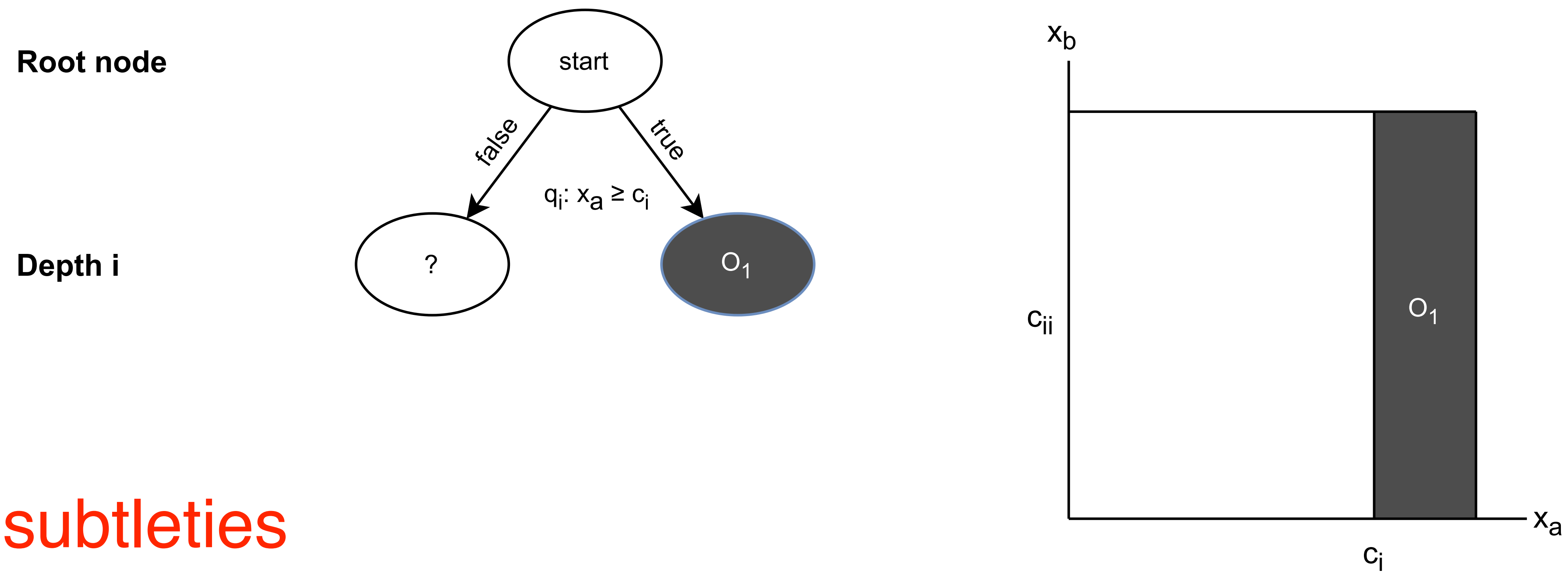
- Advantages & subtleties
  - Cut thresholds & weights determined during training
  - Danger of "memorizing" boundaries (overtraining), so must consider a forest

Decision tree, 2 var example (2)

TM Hong

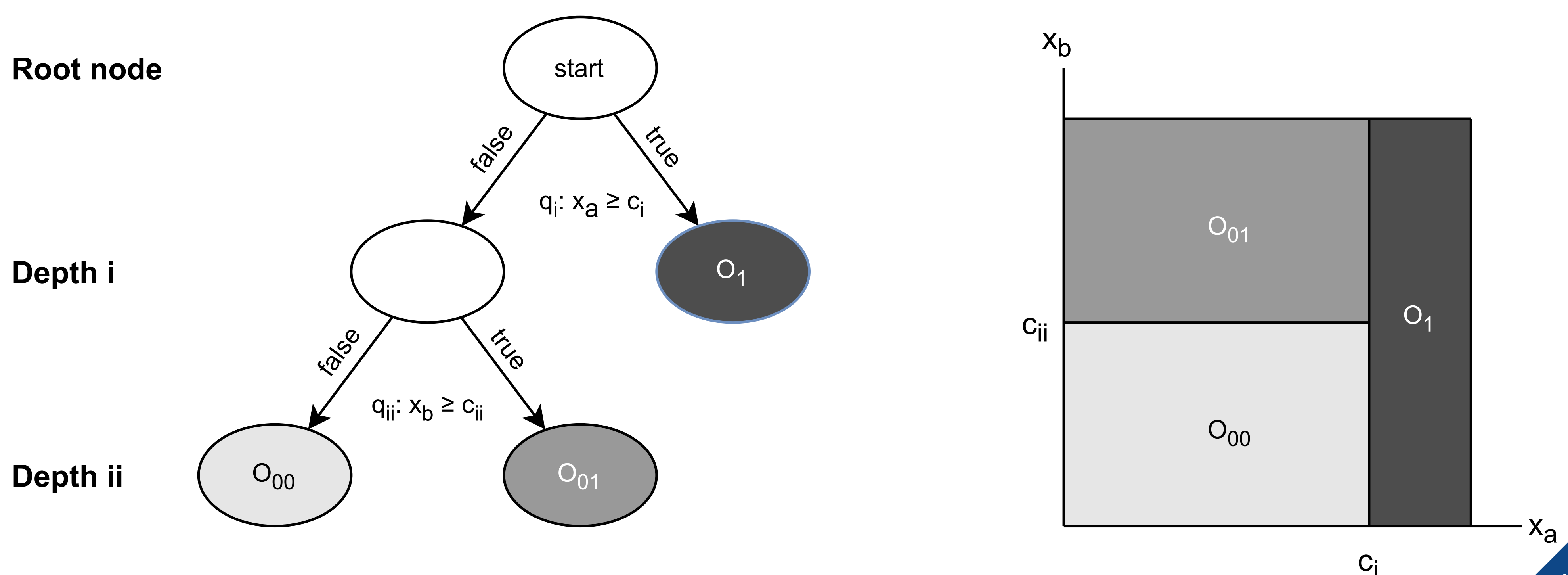
Conventional tree structure | **2d plane: x_a vs. x_b**

First step



- Advantages & subtleties
  - Deterministic, conventional style
  - Cuts in each axis is not independent of each other, so recursive

Full tree

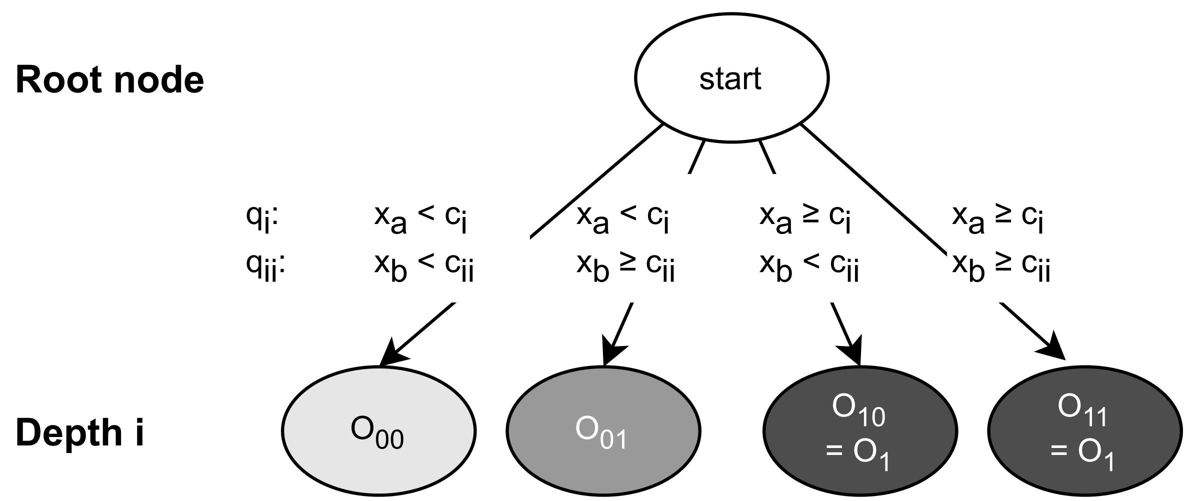


Decision tree, 2 var example (3)

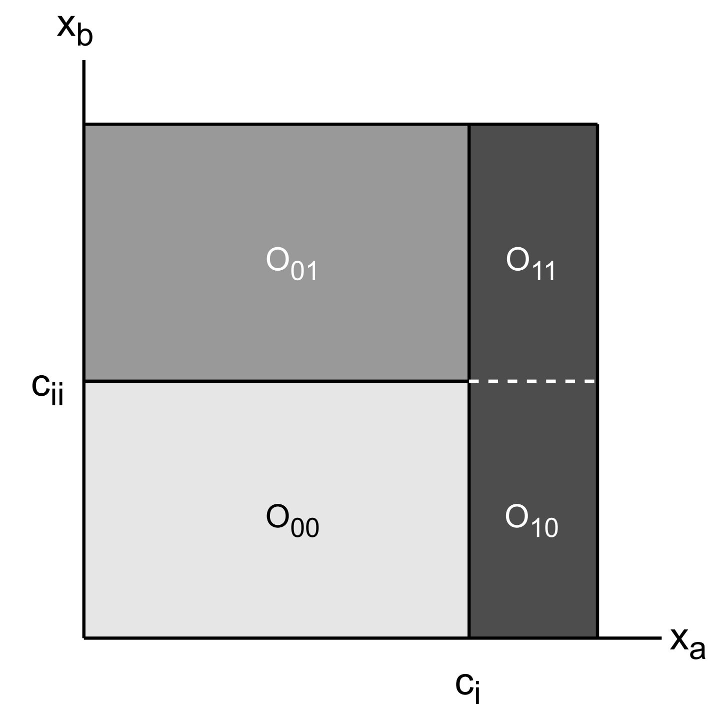
TM Hong

Our approach

Flattened tree

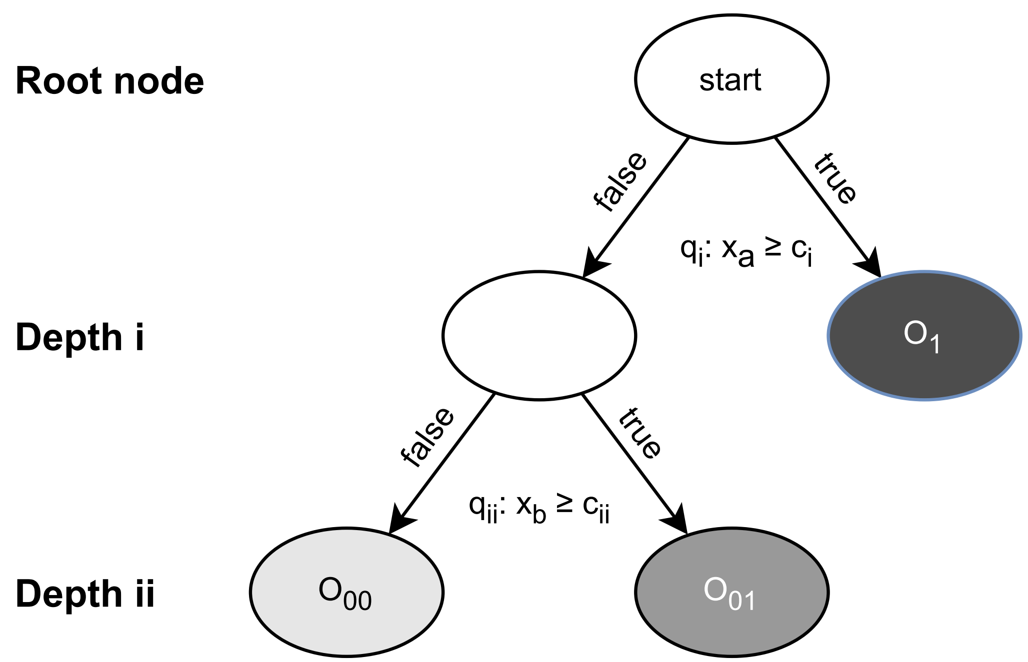


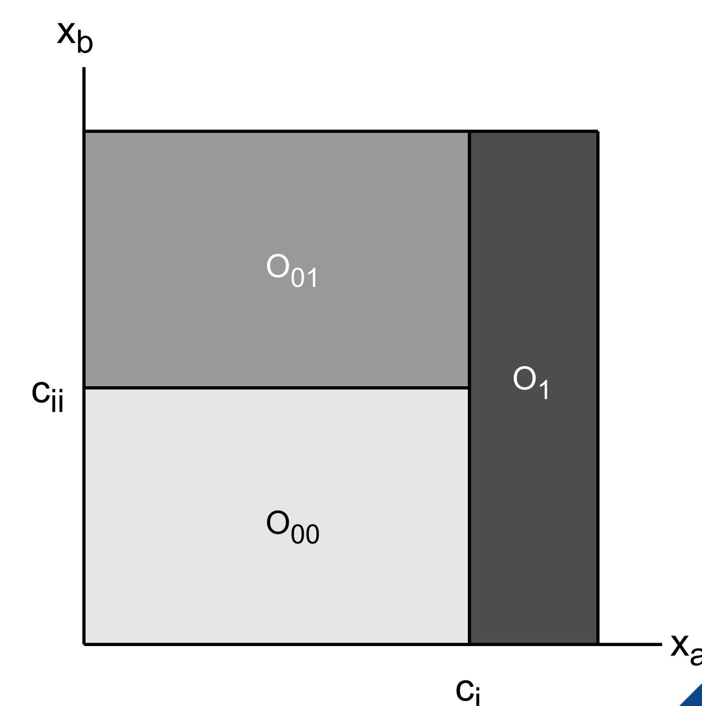
2d plane: x_a vs. x_b



- Advantages & subtleties
  - Each axis is independent of each other → Bin search problem on a grid
  - Does not scale well for very deep trees (but do you really need it at L1?)

Full tree





Forest of boosted decision trees

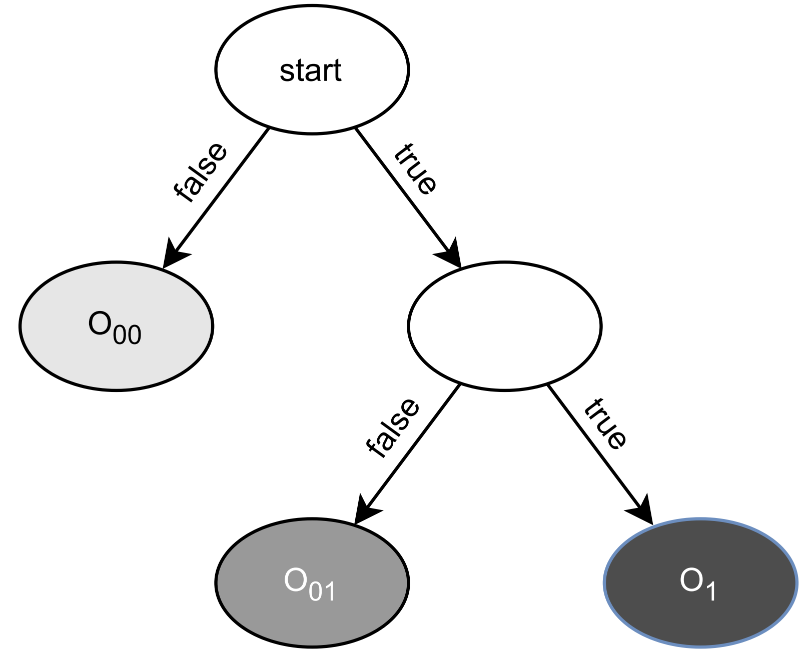
TM Hong

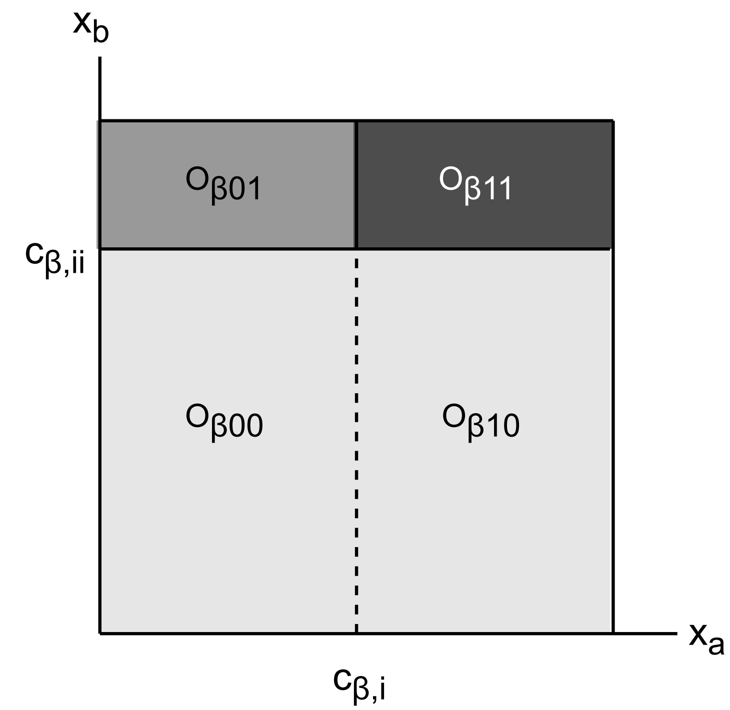
- Advantages & subtleties
  - Use TMVA software to train the BDT (support for other sw coming)

Our approach

2nd tree

Decision tree τ_β with boost weight w_β

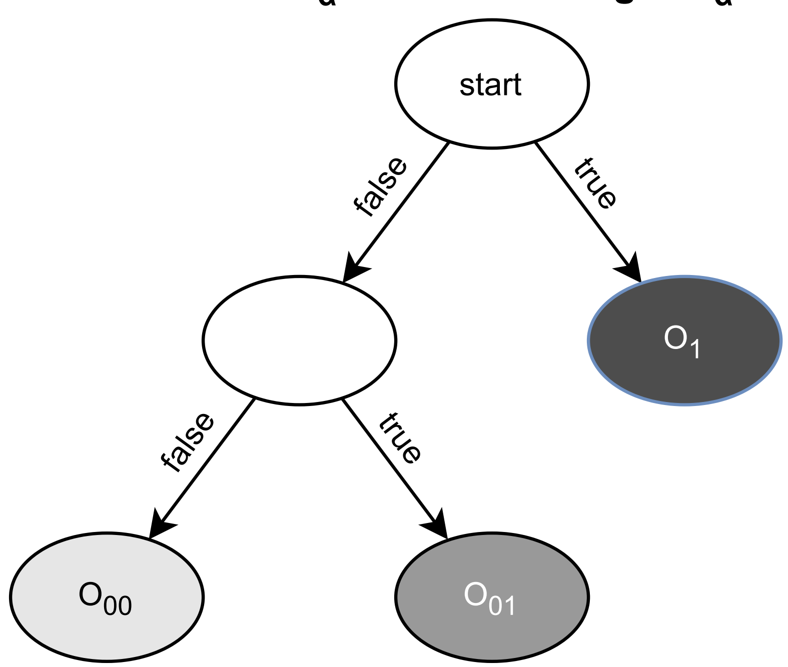


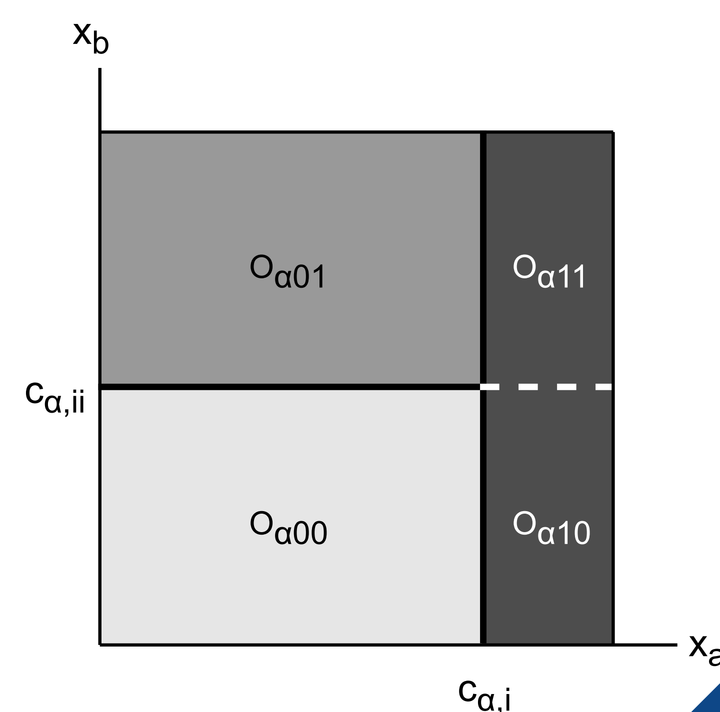


  - Can we pre-merge the trees for firmware? Yes, next slide.

1st tree

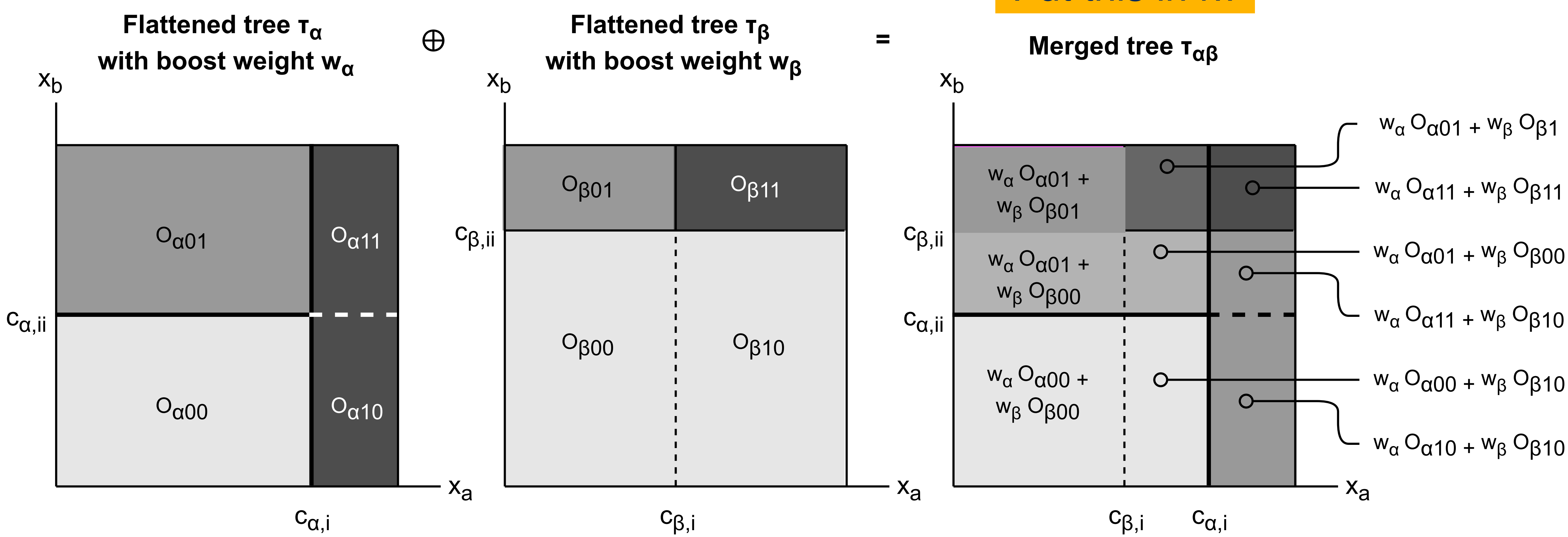
Decision tree τ_α with boost weight w_α





Merging of the forest

Put this in fw



- Advantages & subtleties
  - Merging is pre-processed before implementation in firmware
  - This is using adaptive boosting. Gradient boosting cannot pre-merge, but we have approximations for that method to improve performance.

- Physics impact of flattening & merging
  - None, bec. encodes the entirety of conventional approach
  - Firmware is a giant look-up table problem

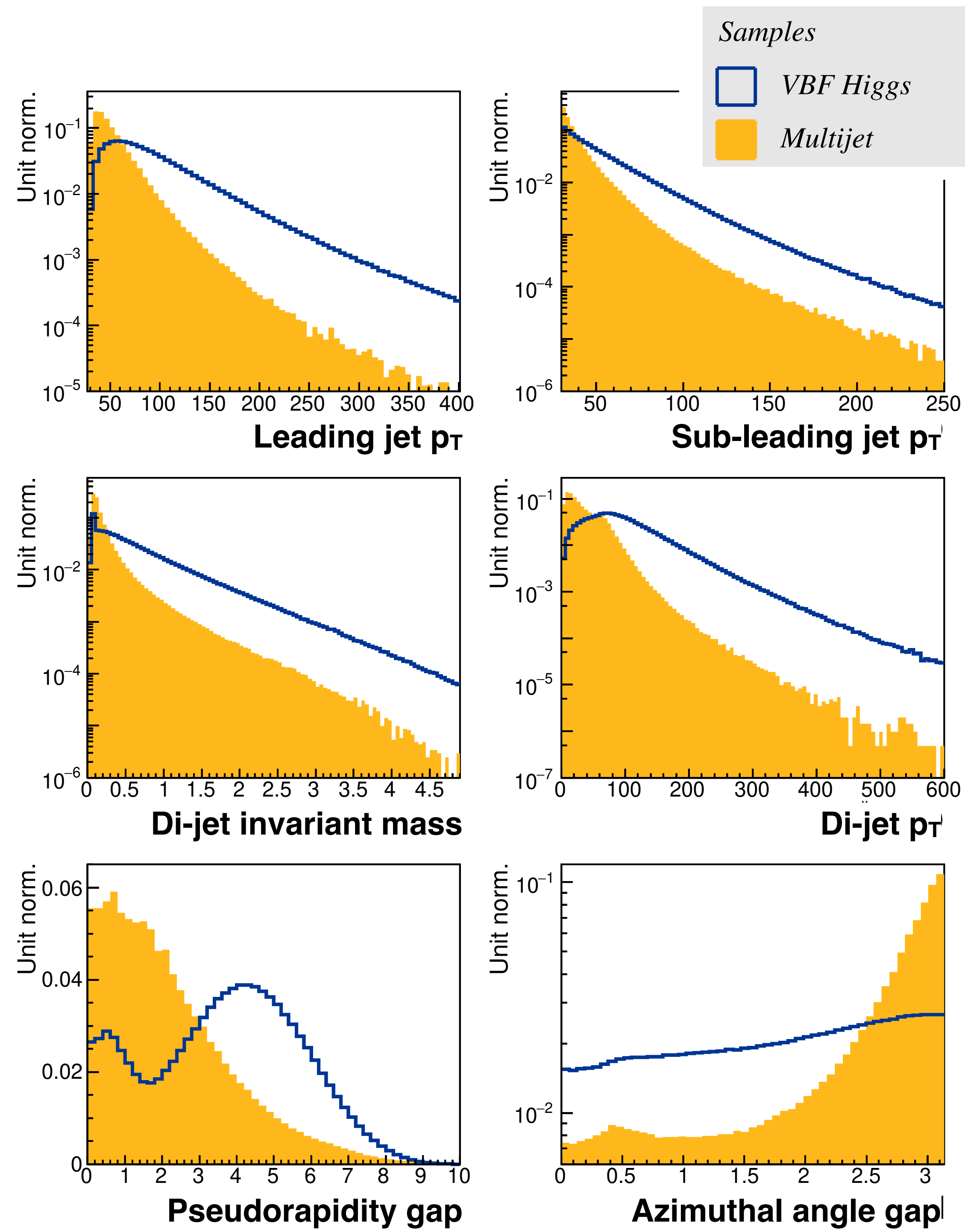
Physics: VBF Higgs vs. multijet

- VBF Higgs vs. Multijet background
  - σ_{Higgs} = 4 pb, two widely separated high-p_T jets
  - σ_{pp} = 80 mb, dominant process at LHC
  - Distributions given on the right

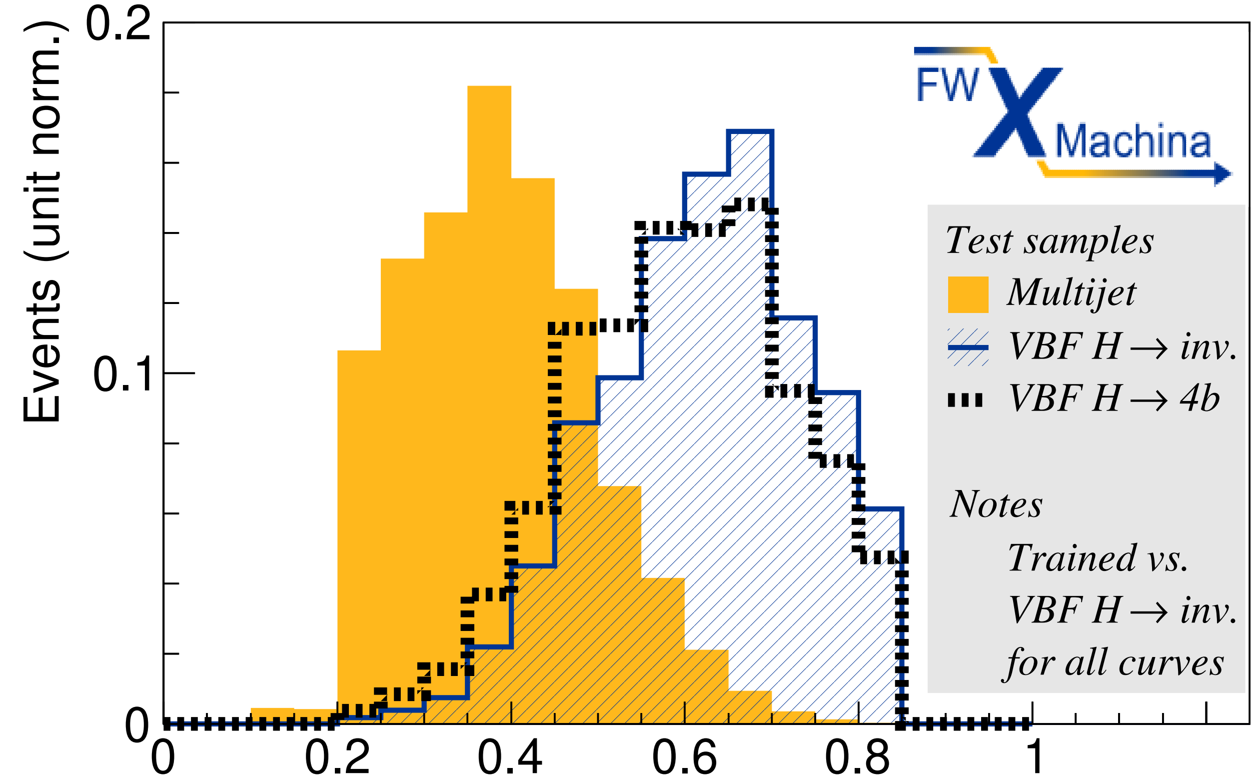
- We consider two decays of the Higgs
  - H → $\nu\bar{\nu}\nu\bar{\nu}$, "invisible"
  - H → $b\bar{b}b\bar{b}$, thru pseudoscalar decays

- Strategy
  - Train BDT to identify VBF jet pair, i.e., train BDT on Multijet vs. VBF H → $\nu\bar{\nu}\nu\bar{\nu}$
  - Apply that BDT to Multijet vs. VBF H → $b\bar{b}b\bar{b}$

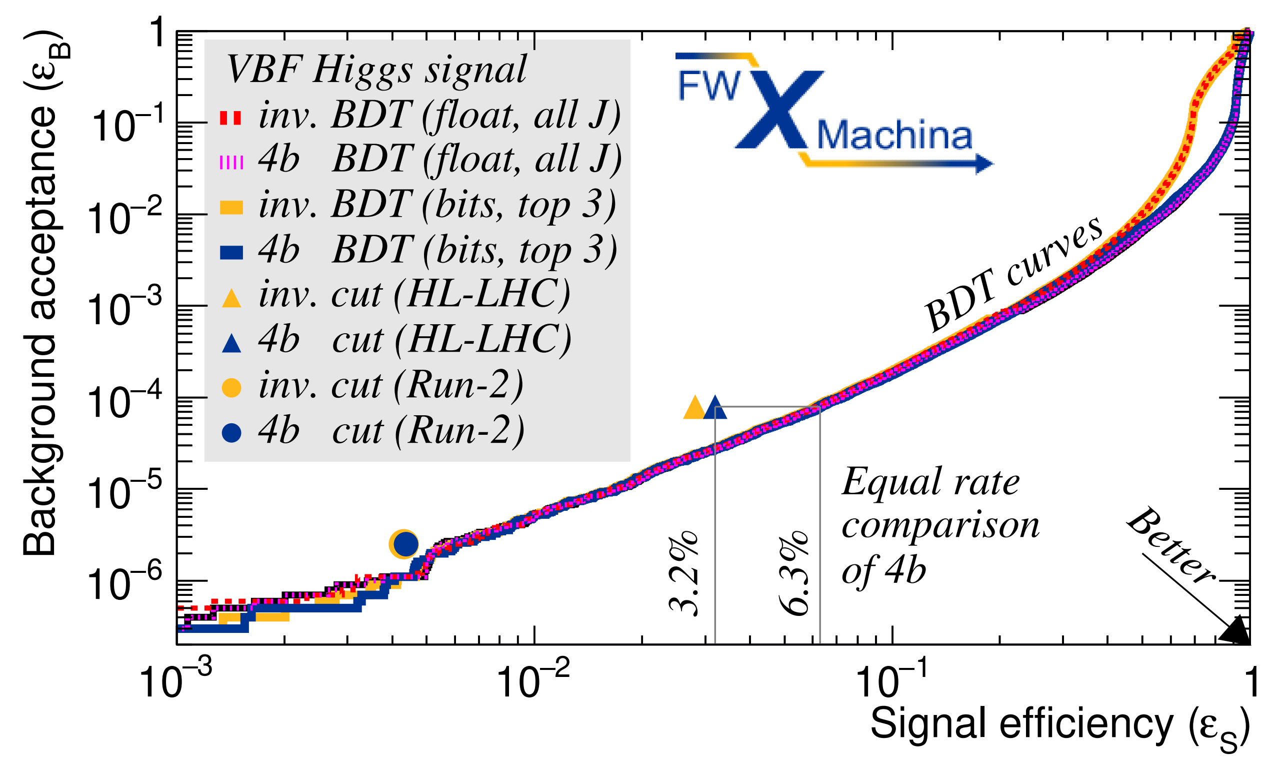
- Why
  - If it works for VBF H → $b\bar{b}b\bar{b}$, then it can be a trigger for VBF independent of the Higgs decay
  - Does it work? Next slide



- It works!
  - Reminder. Did *not* train on VBF H → $b\bar{b}b\bar{b}$
  - Subtlety re: jet selection (see paper)
  - Distributions given on the right



- Performance comparison
  - Try to mimic ATLAS HL-LHC cuts as best we can using Madgraph + Delphes
  - Two-fold signal efficiency improvement from ATLAS-inspired → fwX results



- Details
  - We validated our setup to reproduce the signal efficiency in the ATLAS Run-2 paper
  - Comparison using bit integers, not floats

Firmware: VBF Higgs vs. multijet

- Ran two configurations
  - Optimized version
  - Non-optimized version (for comparison)
  - Both using 100 trees, max depth of 4
  - Results given on the right

- Performance
  - 5 clock ticks = 16 ns
  - Negligible resource usage

- Benchmark using e^+ vs. γ
  - In the paper, we also define one set of parameters to scale up one param. at a time
  - Uses 4 variables, 8 bits & same as above
  - 3 clock ticks = 10 ns
  - Negligible resource usage

	VBF H Optimized	VBF H Non-opt
N_{var}	5	7
$N_{bit\text{-}var}$ $N_{bit\text{-}score}$	8 16	12 16
N_{bin}	40k	1M
Latency	5 ticks	6 ticks
LUT	1%	1.5%
Flip Flops	~0	~0
BRAM	2%	30%
DSP	0	~0

Back up

Comparison to hls4ml

TM Hong

- Details
  - Ideally we would run hls4ml's example & compare, but we can't as-is because they run a 5-class jet identification (b, W, top, g, q)
  - We ran hls4ml on the same dataset with the same configuration as in our paper

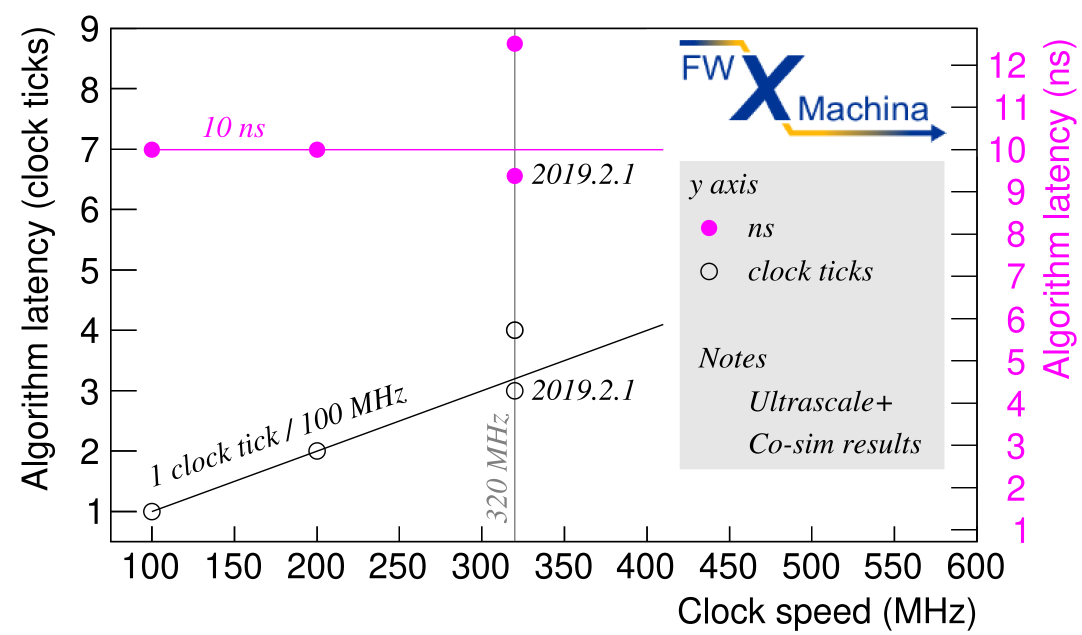
Parameter	FWXMACHINA	hls4ml-Conifer	Comments
ML training setup			
Training software	*TMVA*	*TMVA*	same
Physics problem	electron vs. photon	electron vs. photon	same
Training samples	from ref. [56]	from ref. [56]	same
No. of event classes	2	2	same
No. of training trees	100	100	same
Max. depth	4	4	same
No. of input variables	4	4	See figure 18
Other *TMVA* parameters	*TMVA* defaults	*TMVA* defaults	same
Nanosec. Optimization	Flattened & merged to 10 final trees, without TREE REMOVER or CUT ERASER	N/A	Unique to FWX
FPGA and firmware setup			
Chip family	Xilinx Virtex Ultrascale+	Xilinx Virtex Ultrascale+	same
Chip model	xcvu9p-flga2104-2L-e	xcvu9p-flga2104-2L-e	same
Vivado HLS version	2019.2	2019.2	same
Clock speed, period	320 MHz, 3.125 ns	320 MHz, 3.125 ns	same
Precision	ap_int⟨8⟩	ap_ufixed⟨10, 5⟩	See text
BIN ENGINE	BSBE	N/A	Unique to FWX
FPGA cost			
Latency	3 clock ticks, 9.375 ns	15 clock ticks, 46.875 ns	-
Interval	1 clock tick, 3.125 ns	1 clock tick, 3.125 ns	same
LUT	1903, $< 0.2\%$ of total	2.3 M, 192% of total	See caption
FF	138, $< 0.01\%$ of total	1.1 M, 44% of total	-
BRAM 18k	8, $< 0.2\%$ of total	0	-
URAM	0	0	same
DSP	0	0	same

- Same setup
- Comparison

Benchmark firmware perform'ce

TM Hong

Parameter	Value	Comments
FPGA setup		
Chip family	Xilinx Virtex Ultrascale+	
Chip model	xcvu9p-flga2104-2L-e	
Vivado version	2019.2.1	
Synthesis type	C-Synthesis	
HLS or RTL	HLS	
HLS interface pragma	None	
Clock speed	320 MHz	Clock period is 3.125 ns
ML training configuration		
ML training method	Boosted decision tree	Binary classification
Boost method	Adaptive	AdaBoost with yes/no leaf
No. of event types to classify	2	Signal vs. background
No. of input variables	4	
No. of trees used for training	100	
Maximum tree depth	4	
Nanosecond Optimization configuration		
Bin Engine type	Bit Shift Bin Engine (BSBE)	
No. of bits for input variables	8 bits for each	
No. of bits for cut thresholds	8 bits for each	
No. of bits for BDT output score	8 bits	
No. of trees after merging	10	Tree Merger via ordered list
No. of final trees	10, none removed	Tree Remover by truncation
No. of bins	26132	Cut Eraser not used
FPGA cost		
Latency	3 clock ticks	9.375 ns
Interval	1 clock tick	3.125 ns
Look up tables	1903 out of 1182240	< 0.2% of available
Flip flops	138 out of 2364480	< 0.01% of available
Block RAM	8 out of 4320	< 0.2% of available
Ultra RAM	0 out of 960	-
Digital signal processors	0 out of 6840	-



- 10 ns is independent of clock from 100-320 MHz

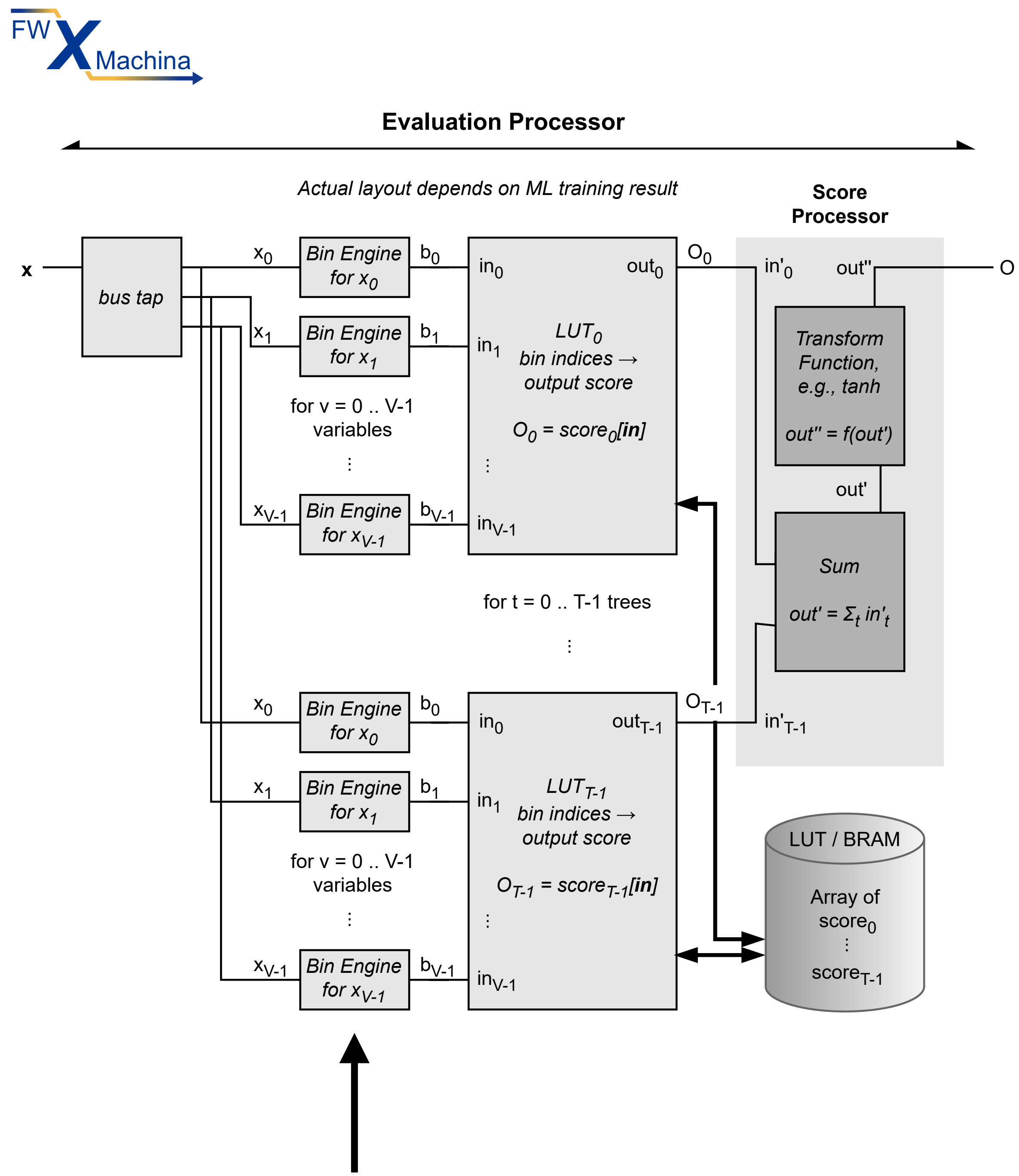
ATLAS-inspired cuts

Table 9: List of input variables for the classification of the VBF Higgs boson vs. multijet process. Also given are the ATLAS-inspired cut-based offline thresholds for Run 2 [64] and HL-LHC [65]. For Run-2, differences arise with respect to the document when the m_{jj} threshold is quoted as 1100 GeV for `L1 MJJ-500-NFF`; we use the $> 99\%$ offline efficiency point, which is achieved around $m_{jj} > 1300$ GeV. for others the offline thresholds are used. For HL-LHC, the single-level scheme values are quoted. The performance of the cut-based approach using these values is compared the performance to the BDT result in figure 16. The non-optimized (non-opt) configuration includes the five variables from the optimized configuration.

Input variable	Description	ATLAS Run-2 offline cut [64], see caption	ATLAS HL-LHC offline cut [65], see caption	Used in BDT
p_{T1}	Leading jet p_{T}	> 90 GeV	> 75 GeV	-
p_{T2}	Subleading jet p_{T}	> 80 GeV	> 75 GeV	Optimized
p_{T12}	Sum $p_{\mathrm{T1}} + p_{\mathrm{T2}}$	-	-	Optimized
$\lvert\eta_1\rvert$	Leading jet η	< 3.2	-	-
$\lvert\eta_2\rvert$	Subleading jet η	< 4.9	-	-
$\prod_\eta$	Product $\eta_1 \cdot \eta_2$	-	-	Optimized
$\lvert\Delta\eta\rvert$	Separation in $\lvert\eta_2 - \eta_1\rvert$	> 4.0	> 2.5	-
$\lvert\Delta\phi\rvert$	Separation in $\lvert\phi_2 - \phi_1\rvert$	< 2.0	< 2.5	non-opt
$\lvert\Delta R\rvert$	$\sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$	-	-	non-opt
m_{jj}	Dijet invariant mass	> 1300 GeV	-	Optimized
p_T^{jj}	Dijet p_{T}	-	-	Optimized

Data flow

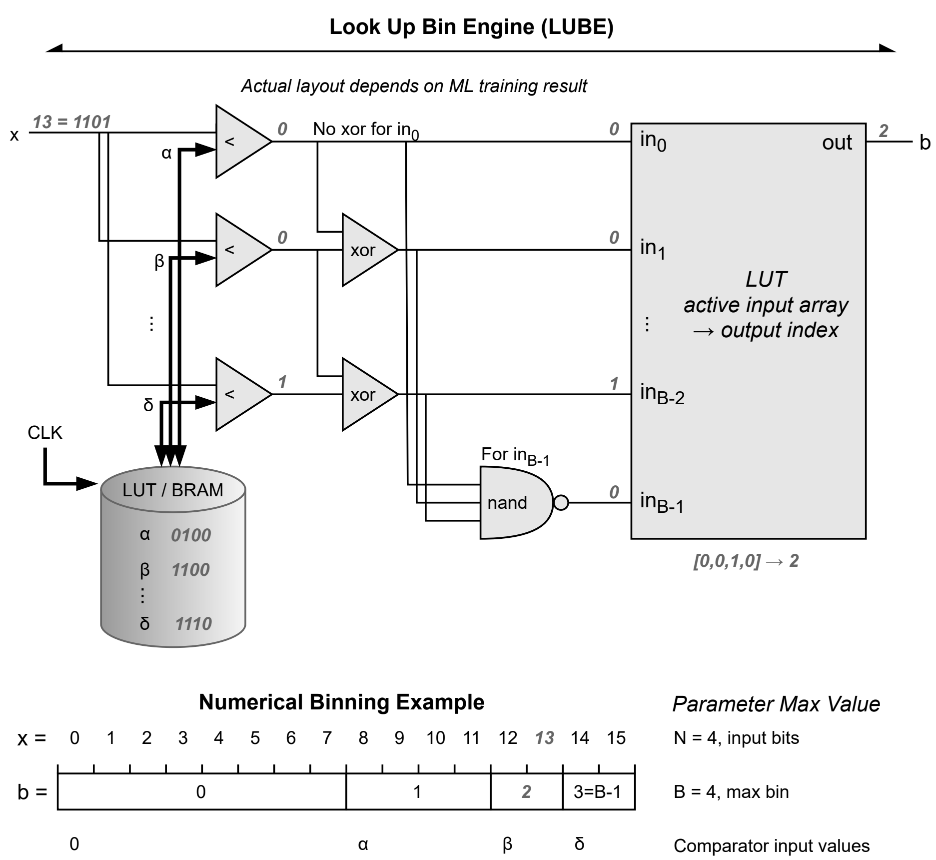
TM Hong

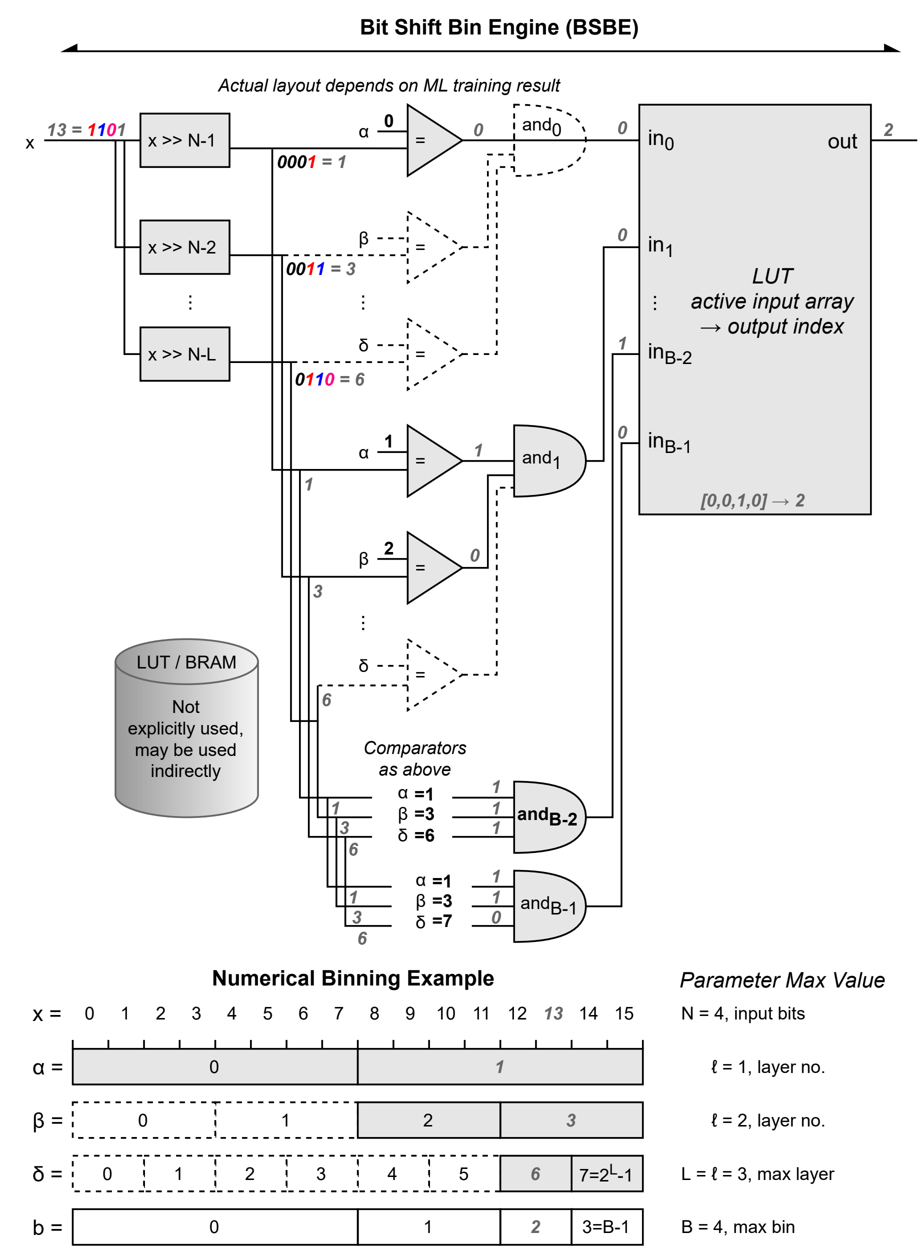


- Each variable is processed independently of each other

Firmware design: Bin Engines

TM Hong





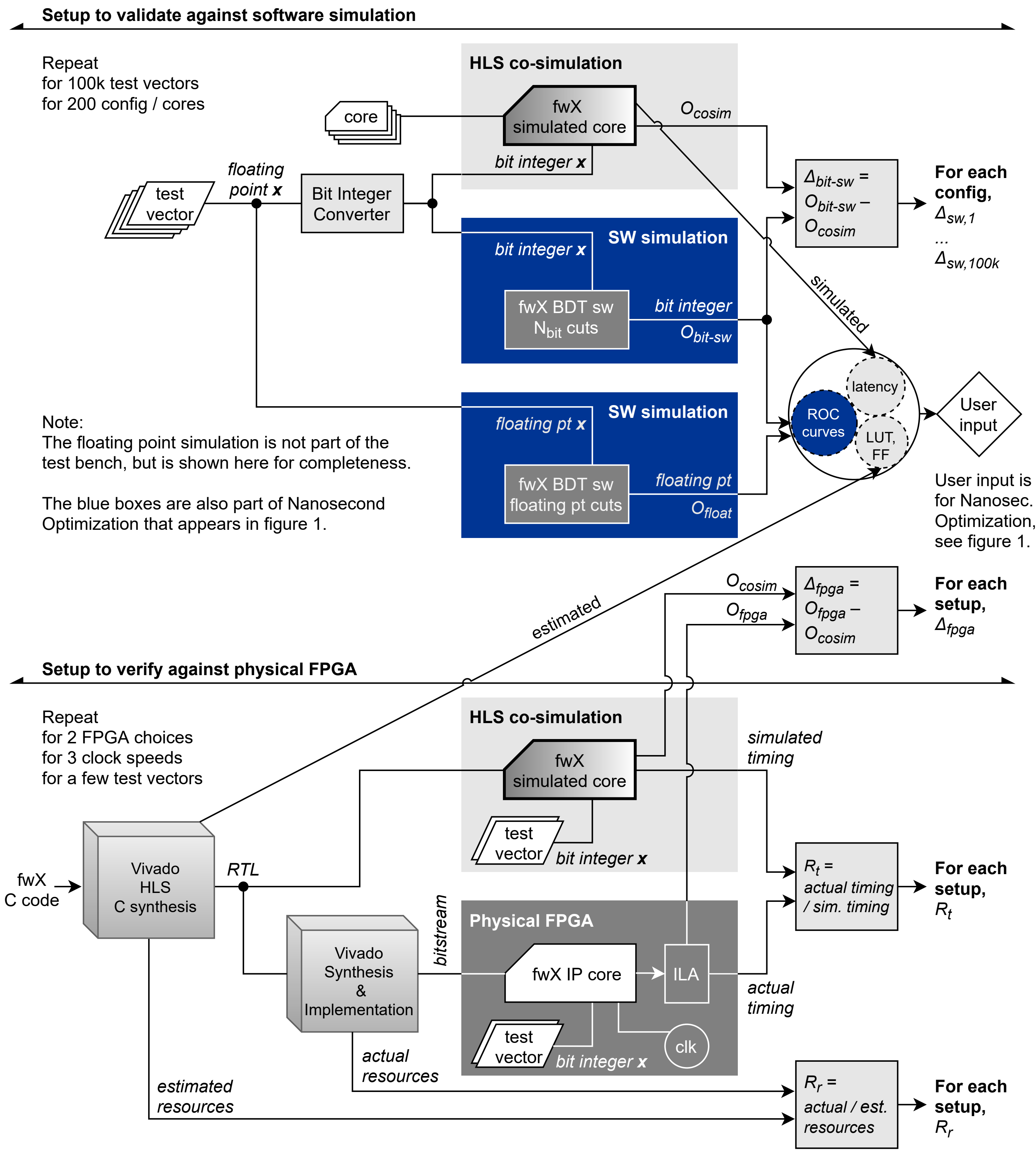
- Look up thresholds in memory, compare

- Bit shift to localize data
  - This is *fast*

- Use combinatoric logic as much as possible without multiplication. No explicit clocked operations.

Test bench

TM Hong



- No difference seen wrt software implementation

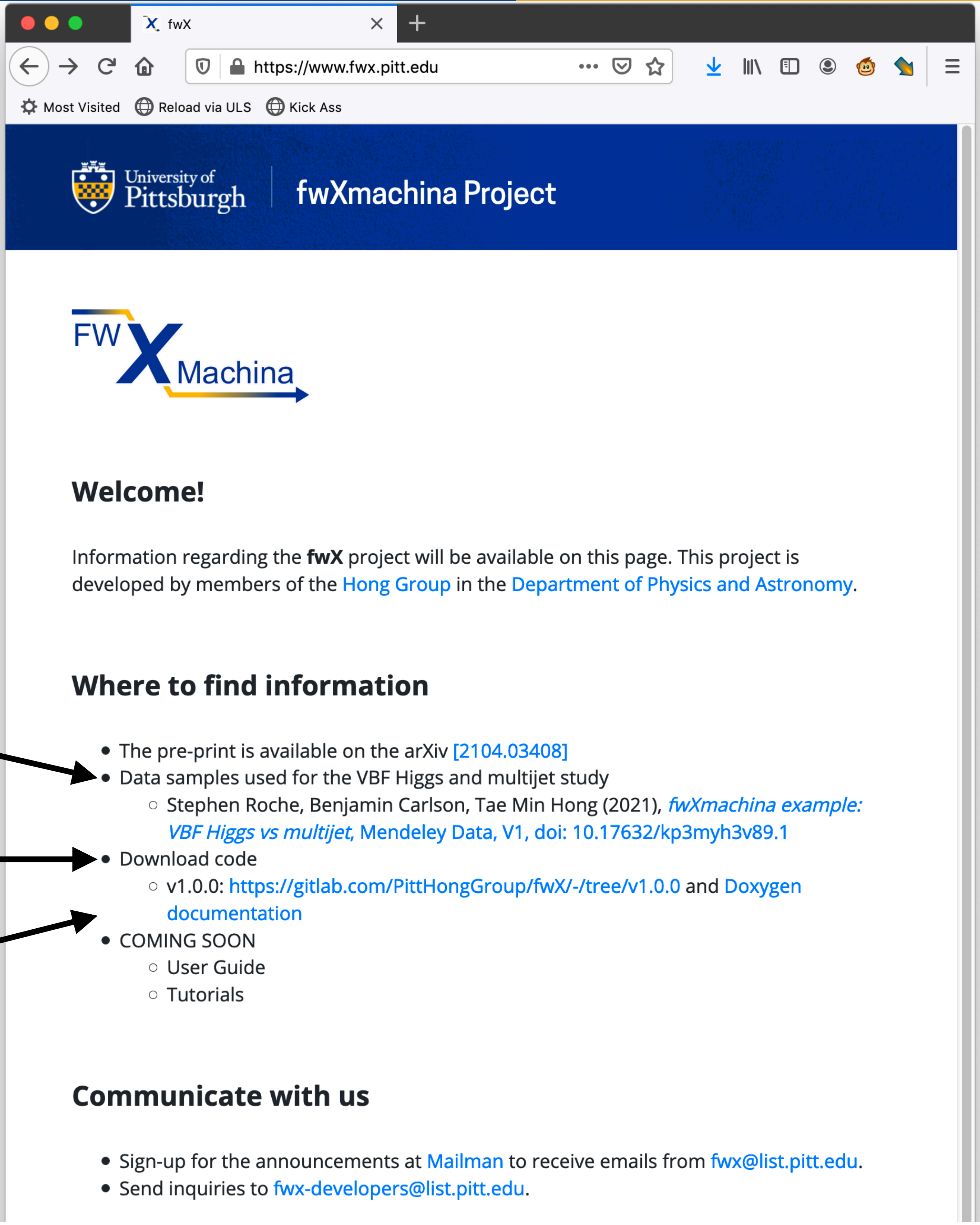
More info

All info at
https://fwx.pitt.edu

data set

git repo

doxygen


fwX
https://www.fwx.pitt.edu
Most Visited Reload via ULS Kick Ass

University of Pittsburgh | fwXmachina Project

FW X Machina

Welcome!

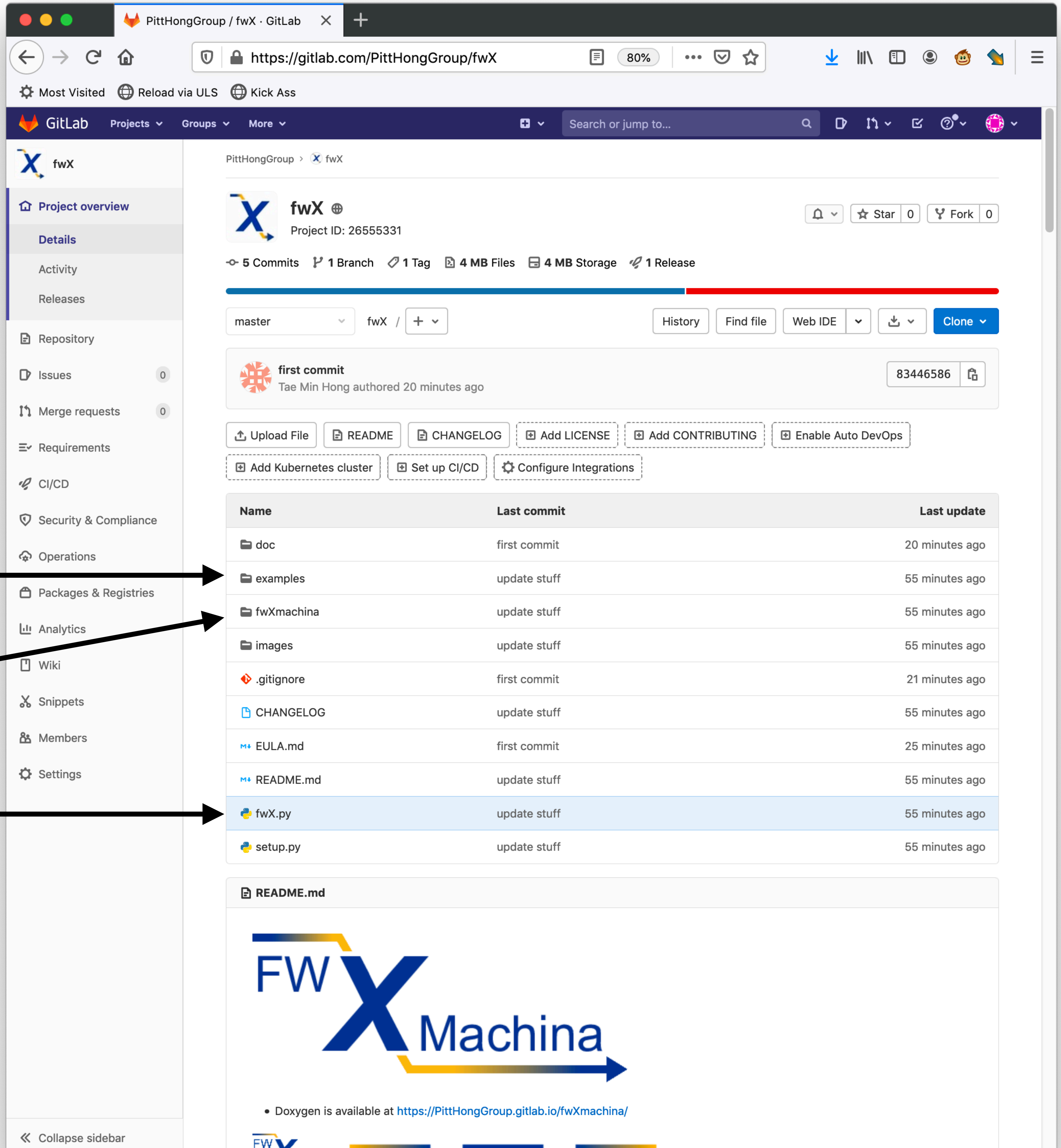
Information regarding the **fwX** project will be available on this page. This project is developed by members of the Hong Group in the Department of Physics and Astronomy.

Where to find information

- The pre-print is available on the arXiv [2104.03408]
- Data samples used for the VBF Higgs and multijet study
  - Stephen Roche, Benjamin Carlson, Tae Min Hong (2021), *fwXmachina example: VBF Higgs vs multijet*, Mendeley Data, V1, doi: 10.17632/kp3myh3v89.1
- Download code
  - v1.0.0: https://gitlab.com/PittHongGroup/fwX/-/tree/v1.0.0 and Doxygen documentation
- COMING SOON
  - User Guide
  - Tutorials

Communicate with us

- Sign-up for the announcements at Mailman to receive emails from fwx@list.pitt.edu.
- Send inquiries to fwx-developers@list.pitt.edu.

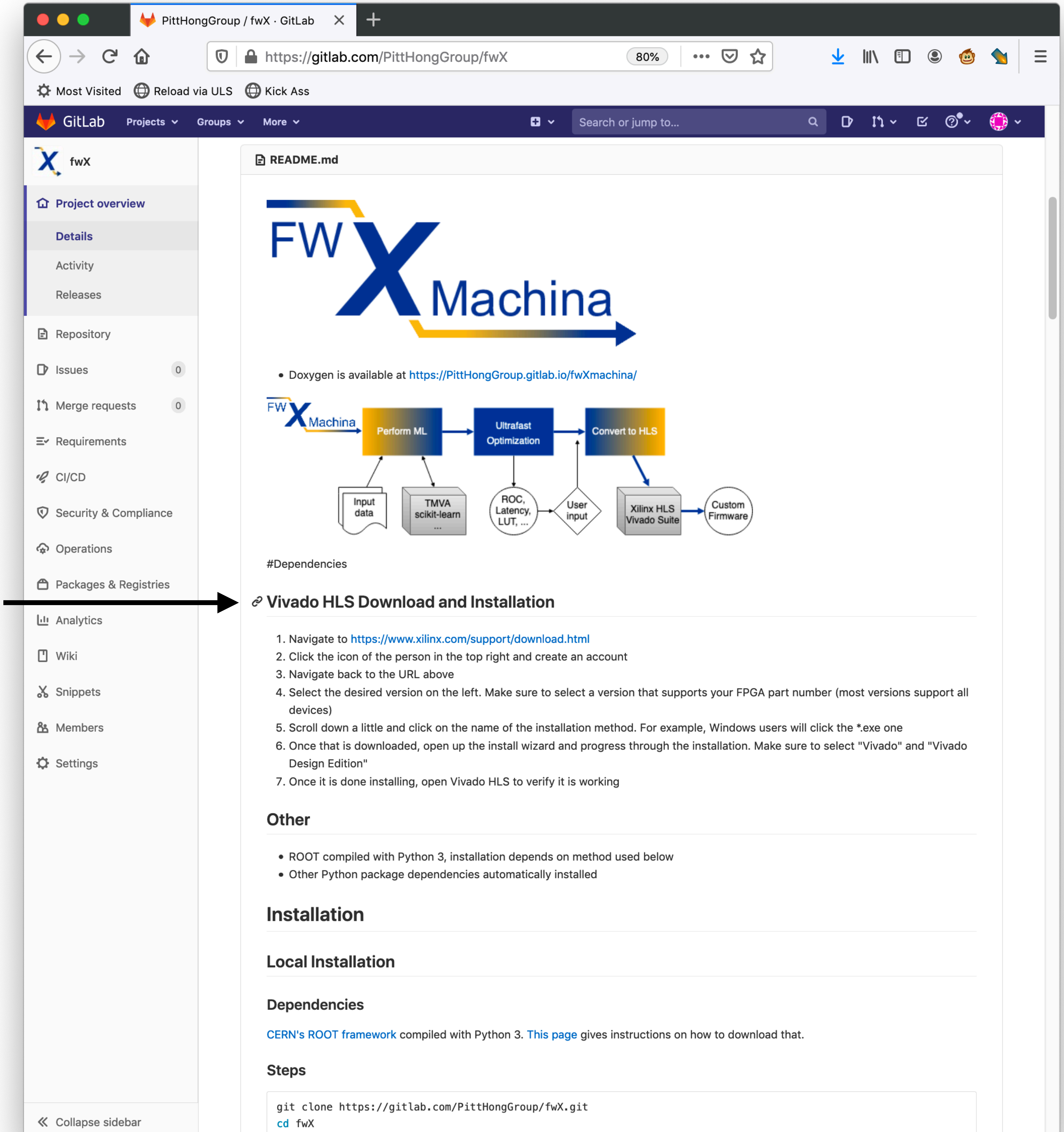


examples

code

driver file

install



- Doxygen is available at https://PittHongGroup.gitlab.io/fwXmachina/

#Dependencies

Vivado HLS Download and Installation

1. Navigate to https://www.xilinx.com/support/download.html
2. Click the icon of the person in the top right and create an account
3. Navigate back to the URL above
4. Select the desired version on the left. Make sure to select a version that supports your FPGA part number (most versions support all devices)
5. Scroll down a little and click on the name of the installation method. For example, Windows users will click the *.exe one
6. Once that is downloaded, open up the install wizard and progress through the installation. Make sure to select "Vivado" and "Vivado Design Edition"
7. Once it is done installing, open Vivado HLS to verify it is working

Other

- ROOT compiled with Python 3, installation depends on method used below
- Other Python package dependencies automatically installed

Installation

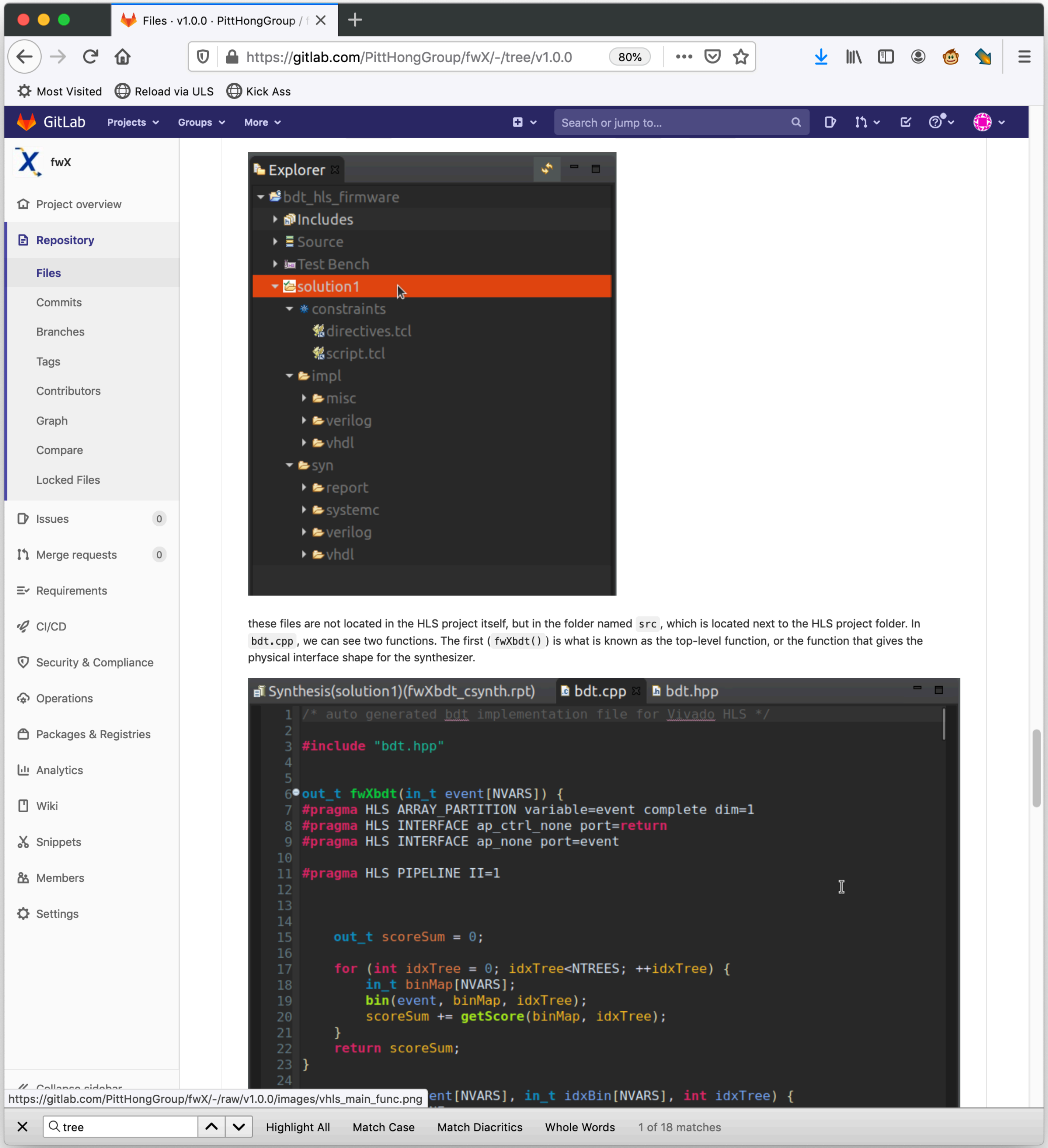
Local Installation

Dependencies

CERN's ROOT framework compiled with Python 3. This page gives instructions on how to download that.

Steps

```
git clone https://gitlab.com/PittHongGroup/fwX.git
cd fwX
pip3 install -e .  # reads the file setup.py to install dependencies and add to pythonpath. The -e flag marks it
```


Files · v1.0.0 · PittHongGroup
https://gitlab.com/PittHongGroup/fwX/-/tree/v1.0.0
fwX
Project overview
Repository
Files
Commits
Branches
Tags
Contributors
Graph
Compare
Locked Files
Issues 0
Merge requests 0
Requirements
CI/CD
Security & Compliance
Operations
Packages & Registries
Analytics
Wiki
Snippets
Members
Settings
Explorer
bdt_hls_firmware
Includes
Source
Test Bench
solution1
constraints
directives.tcl
script.tcl
impl
misc
verilog
vhdl
syn
report
systemc
verilog
vhdl
these files are not located in the HLS project itself, but in the folder named src, which is located next to the HLS project folder. In bdt.cpp, we can see two functions. The first (fwXbdt()) is what is known as the top-level function, or the function that gives the physical interface shape for the synthesizer.
Synthesis(solution1)(fwXbdt_csynth.rpt)
bdt.cpp
bdt.hpp
/* auto generated bdt implementation file for Vivado HLS */
#include "bdt.hpp"
out_t fwXbdt(in_t event[NVARS]) {
#pragma HLS ARRAY_PARTITION variable=event complete dim=1
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE ap_none port=event
#pragma HLS PIPELINE II=1
out_t scoreSum = 0;
for (int idxTree = 0; idxTree<NTREES; ++idxTree) {
in_t binMap[NVARS];
bin(event, binMap, idxTree);
scoreSum += getScore(binMap, idxTree);
}
return scoreSum;
}
https://gitlab.com/PittHongGroup/fwX/-/raw/v1.0.0/images/vhls_main_func.png
tree
Highlight All
Match Case
Match Diacritics
Whole Words
1 of 18 matches

Code documentation

TM Hong