

# GitLab and CI for Vivado Projects

Chaowaroj (Max) Wanotayaroj

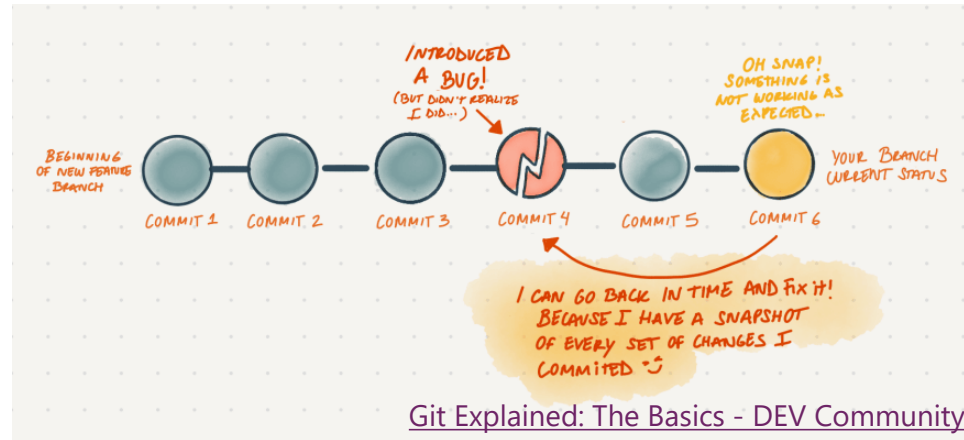
Makoto Tomoto

Yasuyuki Okumura

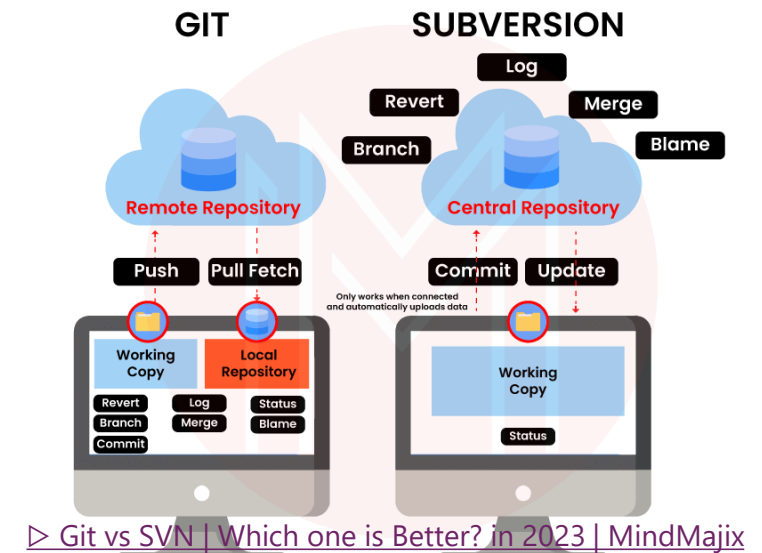


# Introduction – Git

- For code development, we want to keep track of changes in each version

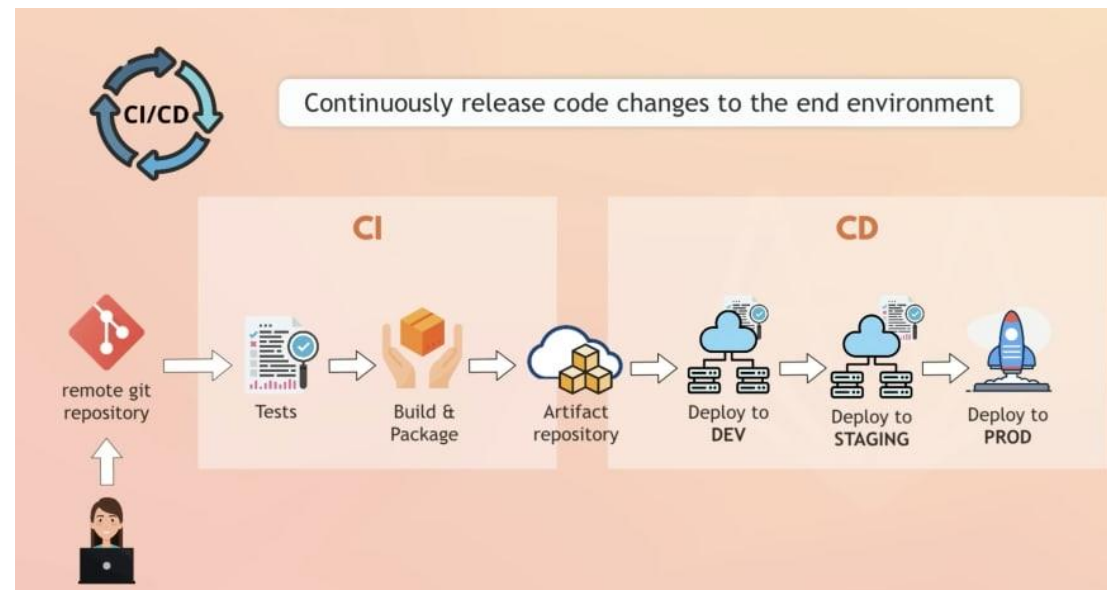


- In the past decade, Git has emerged as the standard for **version control system**
  - Replacing the Subversion (SVN)
- Git is a distributed version control system
  - Full code+history on your machine
  - Faster for most operations



# Introduction - Git vs. GitHub

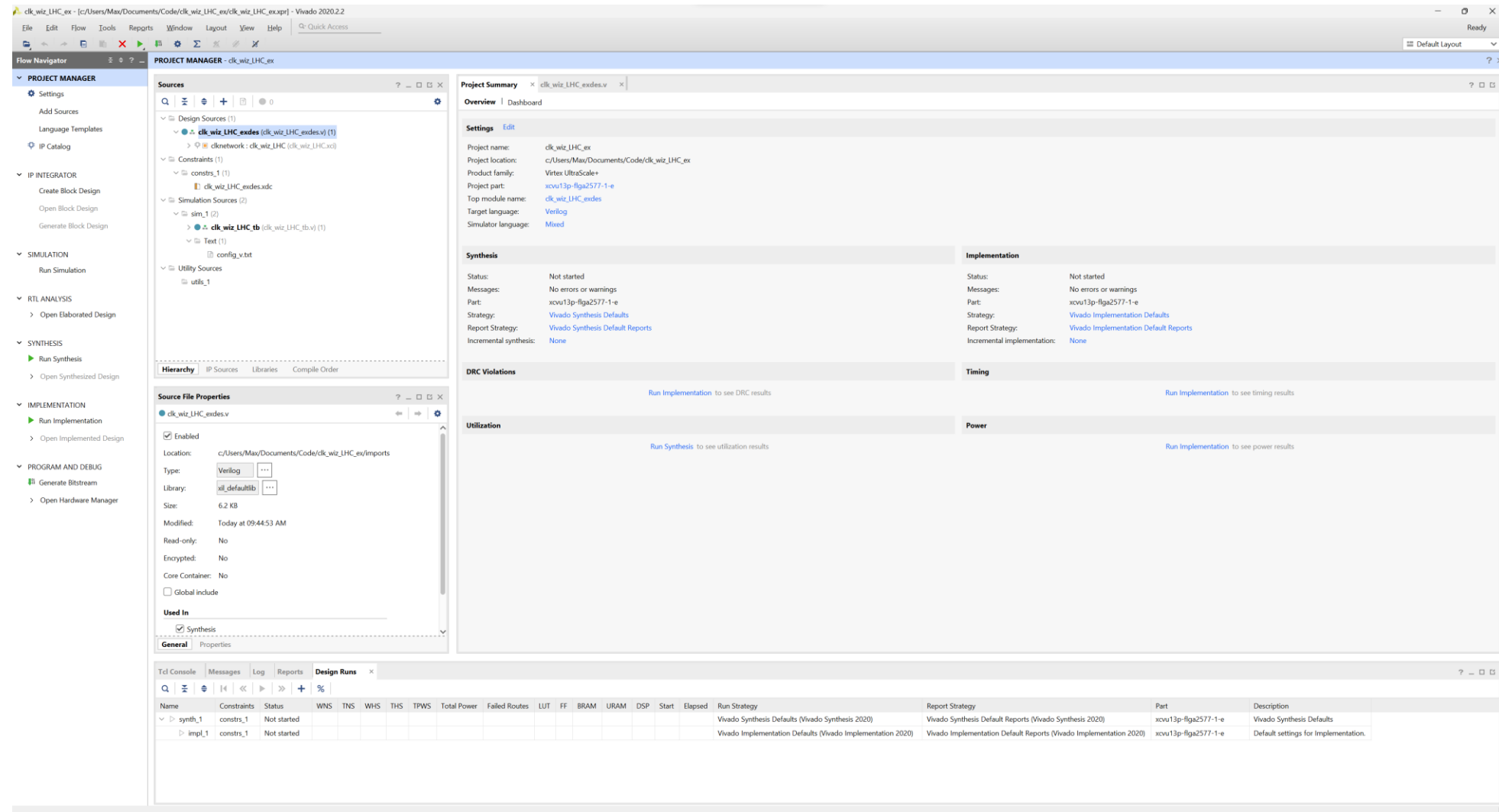
- A service such as GitHub or GitLab let you host a Git repository
- And offer other features on top of it
  - Bug tracking
  - Automate building (compiling, etc.)
- GitLab CI/CD: continuous integration and continuous delivery
  - CI: Test&Build
  - CD: Deploy



# Vivado IDE

# Vivado IDE

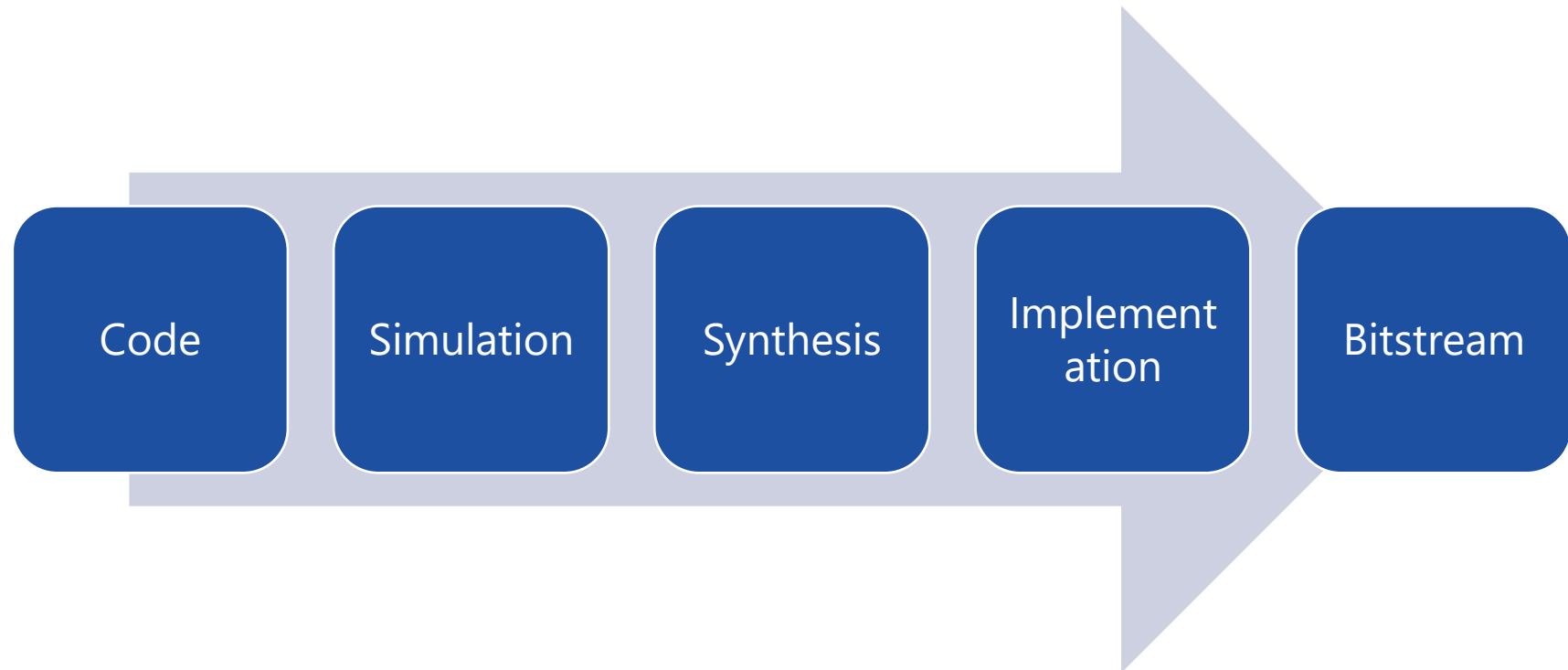
- Vivado is an IDE (Integrated Design Environment) by Xilinx/AMD for FPGA firmware design



# Design Flow

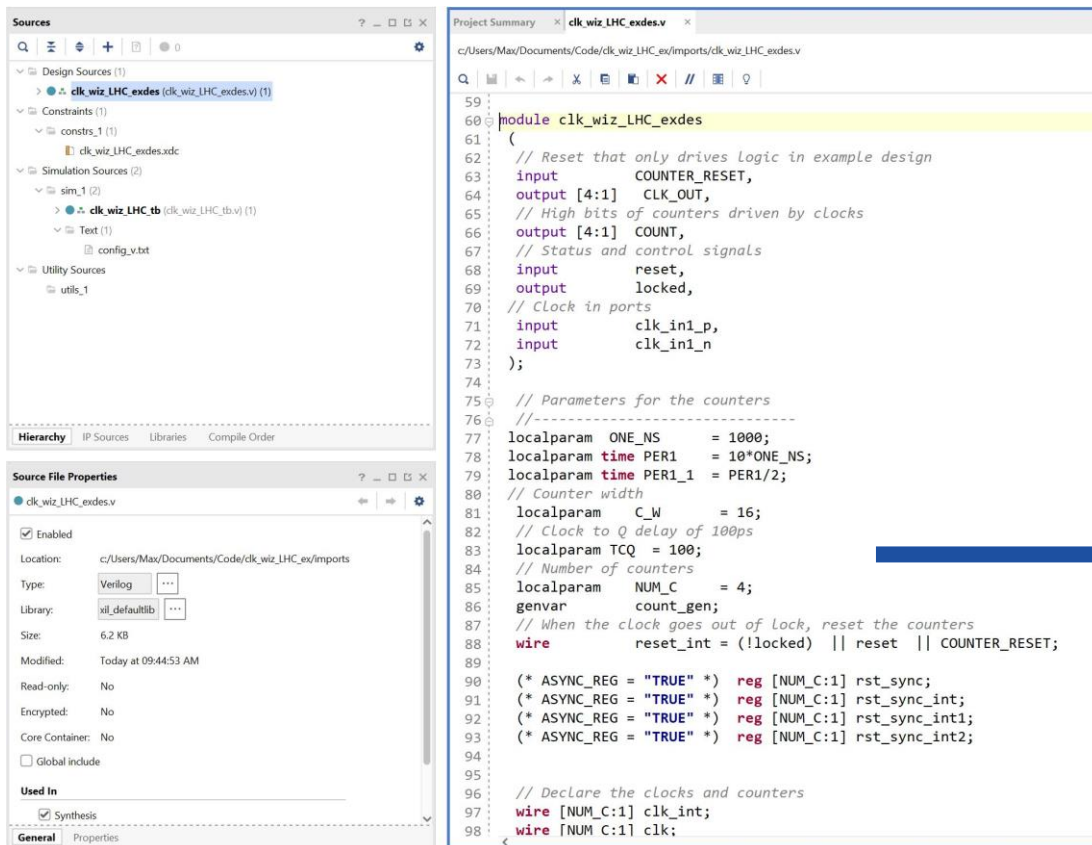
- Typical design flow for a FPGA firmware

- ▼ PROJECT MANAGER
  - ⚙ Settings
  - Add Sources
  - Language Templates
  - 🔗 IP Catalog
- ▼ IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- ▼ SIMULATION
  - Run Simulation
- ▼ RTL ANALYSIS
  - > Open Elaborated Design
- ▼ SYNTHESIS
  - ▶ Run Synthesis
  - > Open Synthesized Design
- ▼ IMPLEMENTATION
  - ▶ Run Implementation
  - > Open Implemented Design
- ▼ PROGRAM AND DEBUG
  - 🔌 Generate Bitstream
  - > Open Hardware Manager



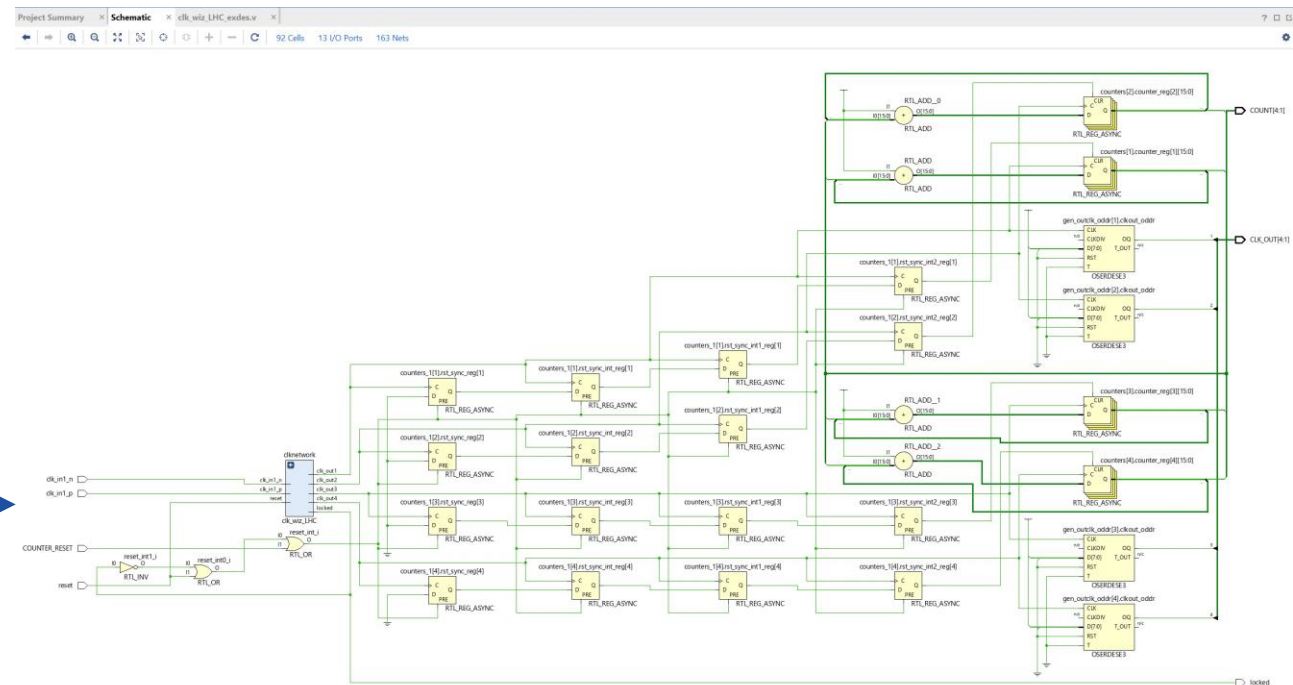
# Code

- Register Transfer Level (RTL) text or block designs
- Constrains
- Schematic: Graphic representation



The screenshot shows a Verilog IDE with two main panes. The left pane displays the 'Sources' tree with files like 'clk\_wiz\_LHC\_exdes.v' and 'clk\_wiz\_LHC\_tb.v'. The right pane shows the Verilog code for 'clk\_wiz\_LHC\_exdes.v'. The code defines a module with inputs for reset, locked, and clock signals, and outputs for counter reset, count, and clock. It includes parameters for counter width and clock delay, and declares registers for asynchronous resets.

```
59
60 module clk_wiz_LHC_exdes
61 (
62     // Reset that only drives logic in example design
63     input    COUNTER_RESET,
64     output [4:1] CLK_OUT,
65     // High bits of counters driven by clocks
66     output [4:1] COUNT,
67     // Status and control signals
68     input    reset,
69     output   locked,
70 // Clock in ports
71     input    clk_in1_p,
72     input    clk_in1_n
73 );
74
75 // Parameters for the counters
76 //-----
77 localparam ONE_NS    = 1000;
78 localparam time PER1  = 10*ONE_NS;
79 localparam time PER1_1 = PER1/2;
80 // Counter width
81 localparam C_W       = 16;
82 // Clock to Q delay of 100ps
83 localparam TCQ       = 100;
84 // Number of counters
85 localparam NUM_C     = 4;
86 genvar    count_gen;
87 // When the clock goes out of lock, reset the counters
88 wire      reset_int = (!locked) || reset || COUNTER_RESET;
89
90 (* ASYNC_REG = "TRUE" *) reg [NUM_C:1] rst_sync;
91 (* ASYNC_REG = "TRUE" *) reg [NUM_C:1] rst_sync_int;
92 (* ASYNC_REG = "TRUE" *) reg [NUM_C:1] rst_sync_int1;
93 (* ASYNC_REG = "TRUE" *) reg [NUM_C:1] rst_sync_int2;
94
95
96 // Declare the clocks and counters
97 wire [NUM_C:1] clk_int;
98 wire [NUM_C:1] clk;
```



# Simulation

- Check behavior of the design

The screenshot displays a behavioral simulation environment with three main panels:

- Scope Table:** Lists variables to be monitored, including `clk_wiz`, `dut`, and `gbl`.
- Objects Table:** Lists the current state of various objects, such as `clk_in1` (Logic, Value 1), `reset` (Logic, Value 1), and `timeout_counter` (Array, Value 0029).
- Waveform Viewer:** Shows a time-based plot of the selected variables. The time axis ranges from 999,984 ps to 1,000,000 ps. The waveform shows digital signals for `clk_in1`, `reset`, and various counters and phase-related signals.



# Synthesis

- “Compile” the design into list of logic gates and other FPGA primitives

The screenshot displays the Xilinx Vivado IDE interface for a synthesized design. The top window shows the 'Netlist' view for the project 'xcvu13p-flga2577-1-e'. The netlist structure is as follows:

- clk\_wiz\_LHC\_exdes
  - Nets (230)
  - Leaf Cells (112)
  - clknetwork (clk\_wiz\_LHC)
    - Nets (8)
    - inst (clk\_wiz\_LHC\_clk\_wiz\_LHC\_clk\_wiz)

The bottom window shows the 'Netlist Properties' for 'clk\_wiz\_LHC\_exdes'. The 'Primitive Statistics' table is as follows:

Primitive type	Count
CLB/CARRY	8
CLB/LUT	6
CLOCK/BUFFER	5
CLOCK/PLL	1
I_O/INPUT_BUFFER	3
I_O/OUTPUT_BUFFER	9
I_O/IOCELL	4

The right window shows the 'Device' view for 'clk\_wiz\_LHC\_exdes.v', displaying a grid of logic cells (X1Y1 to X7Y15) and their connections. The grid is labeled with 'SLR0' through 'SLR9' on the right side, indicating the slice row structure of the device.

# Implementation

- Map the synthesized design to actual hardware

The screenshot displays the Xilinx Vivado implementation tool interface. The top-left pane shows the Netlist hierarchy for 'clk\_wiz\_LHC\_exdes', including Nets (177), Leaf Cells (112), and a clock network (clk\_wiz\_LHC) with its own Nets (8) and an instance (inst). The main workspace shows a timing diagram with a grid of clock signals (X2Y7, X3Y7, X4Y7, X5Y7, X2Y6, X3Y6, X4Y6, X5Y6, X2Y5, X3Y5, X4Y5, X5Y5) and a highlighted signal path circled in orange. The bottom-right pane displays the Design Timing Summary table.

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 2.173 ns	Worst Hold Slack (WHS): 0.055 ns	Worst Pulse Width Slack (WPWS): 0.839 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 140	Total Number of Endpoints: 140	Total Number of Endpoints: 96

**All user specified timing constraints are met.**

# Bitstream

- Bitstream to be programmed on hardware
- Check and debug

The screenshot displays the Xilinx Vivado Hardware Manager interface. On the left, the 'PROJECT MANAGER' sidebar is visible. The main window is titled 'HARDWARE MANAGER - localhost\xilinx\_tcf\Digilent\210292645450A'. It shows a 'Hardware' table with the following entries:

Name	Status
localhost (1)	Connected
xilinx_tcf\Digilent\2102926454...	Open
xc7a100t_0 (2)	Programmed
XADC (System Monitor)	
hw_ila_1 (myila)	Idle

A red arrow points to the 'hw\_ila\_1 (myila)' entry. Below the hardware table, the 'Debug Probe Properties' for 'hcount[6:3]' are shown:

Source: NETLIST  
Type: ILA  
Probe type: Data and Trigger  
Width: 4

The 'Display Name' section shows options for Long name (hcount), Short name (hcount), and Custom name.

The main window also displays a 'Waveform - hw\_ila\_1' window. It shows a table of signal values and a corresponding waveform:

Name	Value
clock_25mhz	0
hcount[6:3]	b
[6]	1
[5]	0
[4]	1
[3]	1

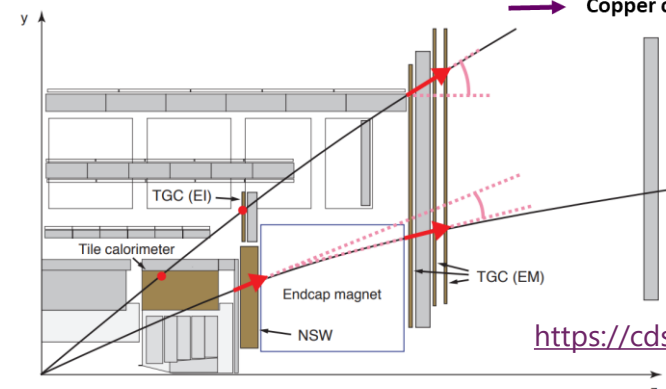
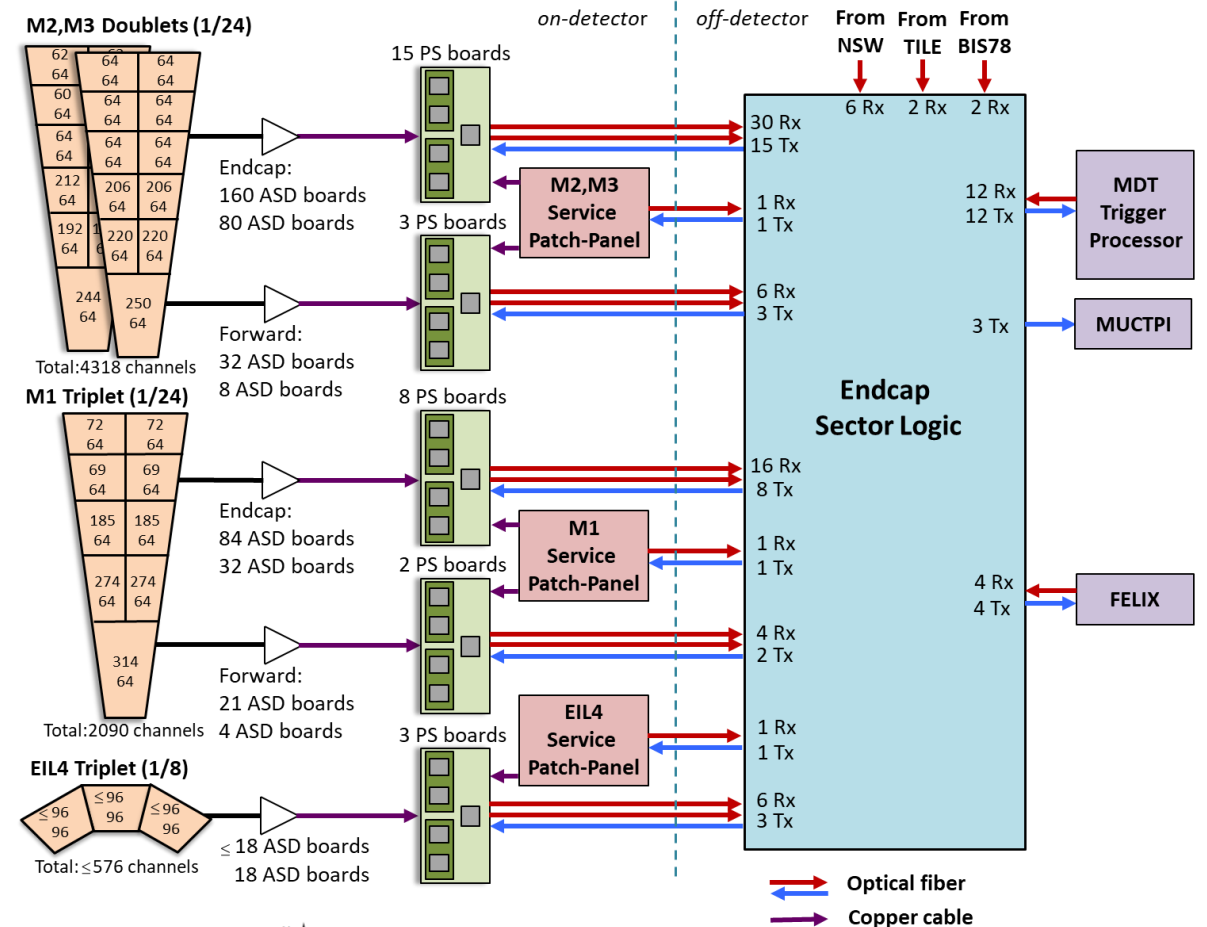
The waveform shows a green signal for 'hcount[6:3]' and a clock signal for 'clock\_25mhz'. A red arrow points to the 'Trigger Setup - hw\_ila\_1' window, which has a '+' button to add probes.

[Integrated Logic Analyzer ILA \(mit.edu\)](https://www.intel.com/programmable/edgearch/ila)

# The Sector Logic (SL) Board

# SL Board

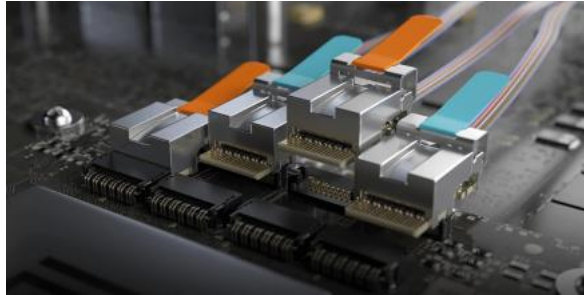
- Develops for TGC Phase-II upgrade
- On the off-detector side of ATLAS
- Gathering and processing data from many parts of the detector
  - One Endcap Sector Logic board covers 1/24 of the Big Wheel TGC (one forward trigger sector and two endcap trigger sectors)
  - Also receive information from NSW and Tile calorimeter
  - Provide L0 MDT Trigger Processor with relevant data and receive back reconstructed tracks
- Sending all hit information to the SL board allow more sophisticated algorithm



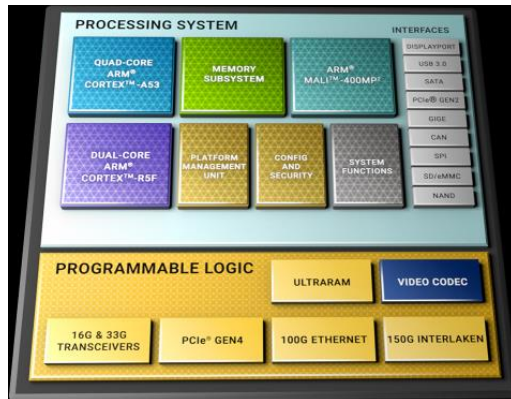
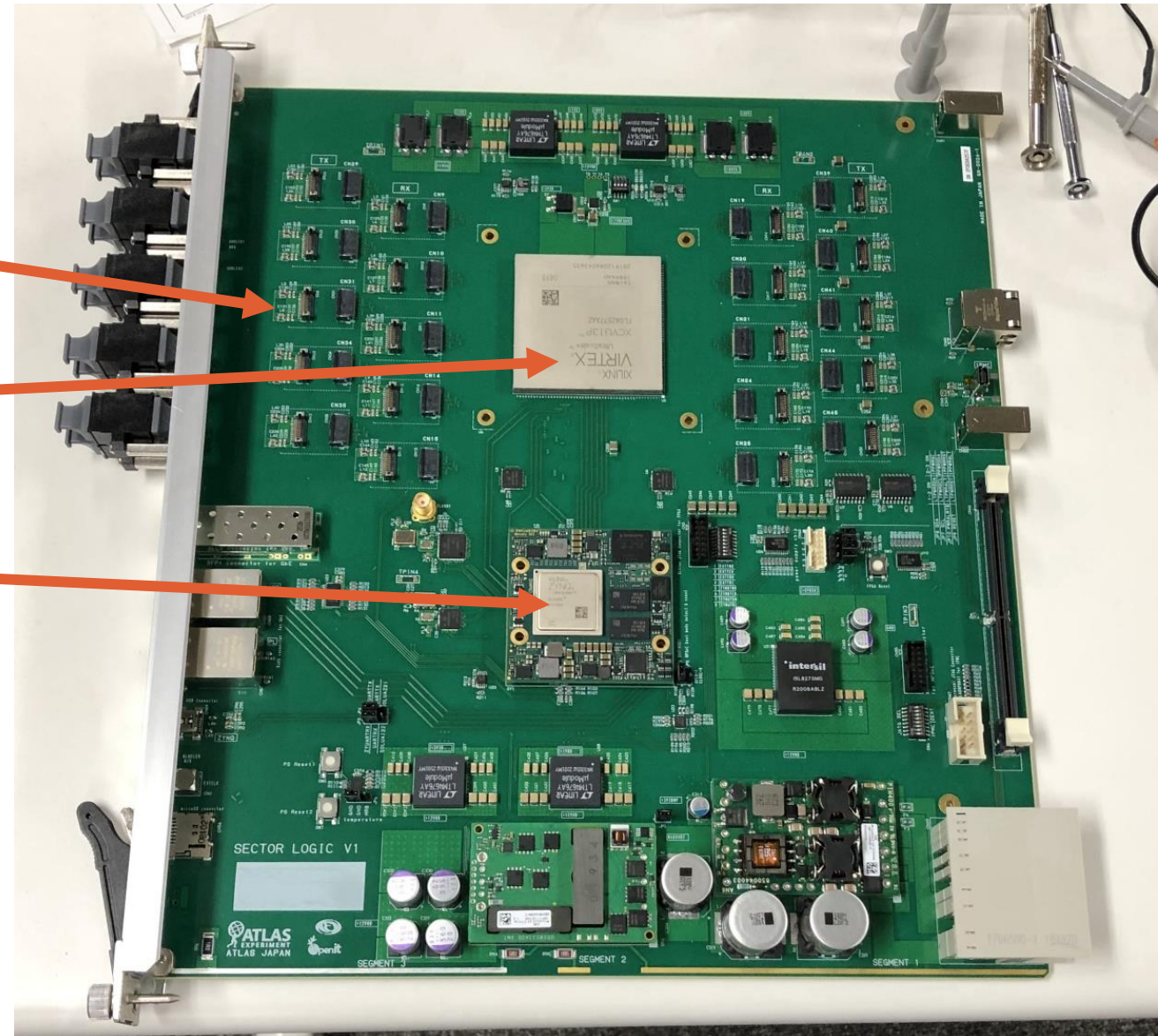


# Hardware (for this talk)

- Firefly connectors (elec->optical transceivers)

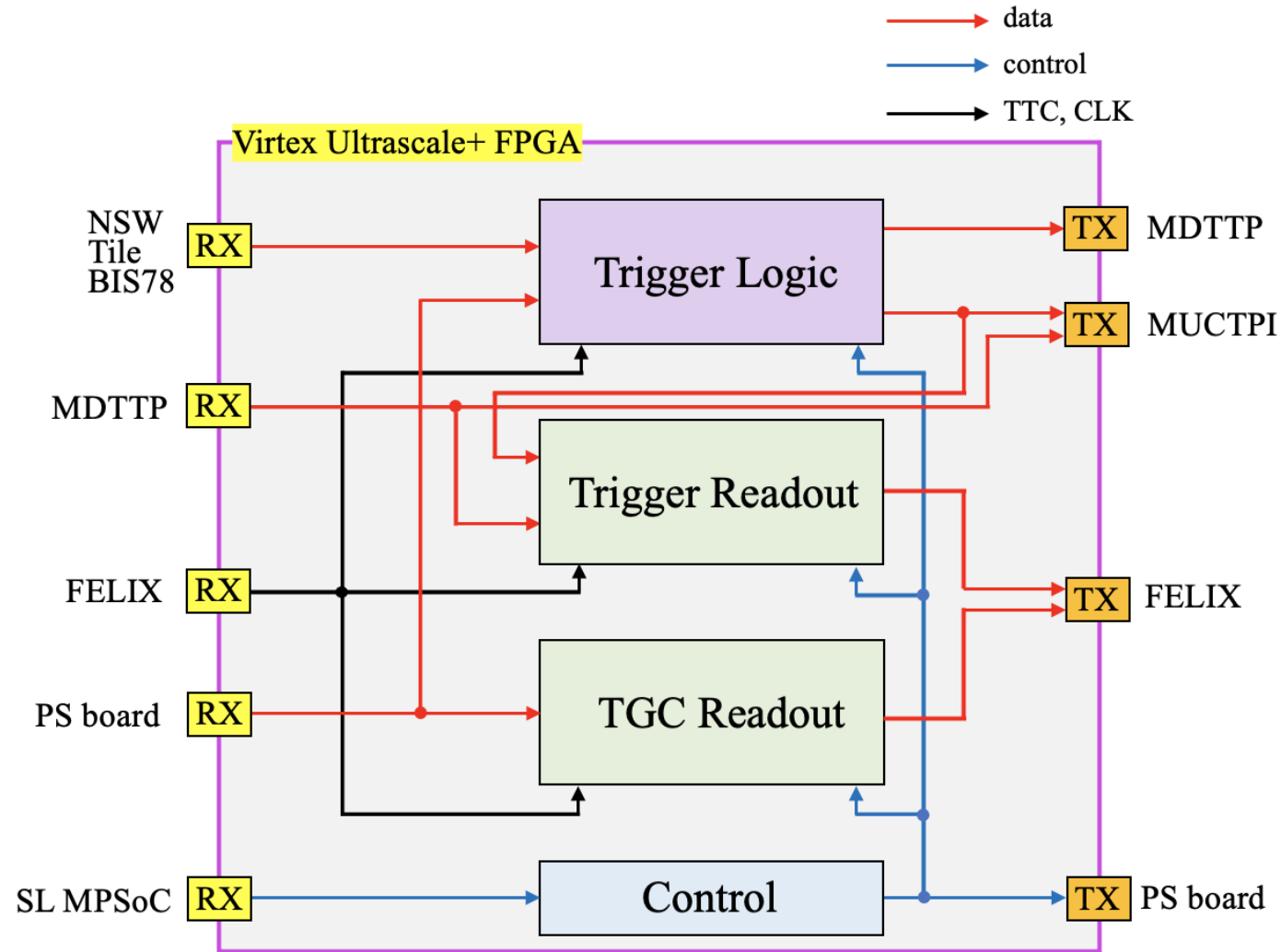


- Virtex Ultrascale+
  - High-performance FPGA
- Zynq Ultrascale+ MPSoC
  - Multiprocessor System on a Chip



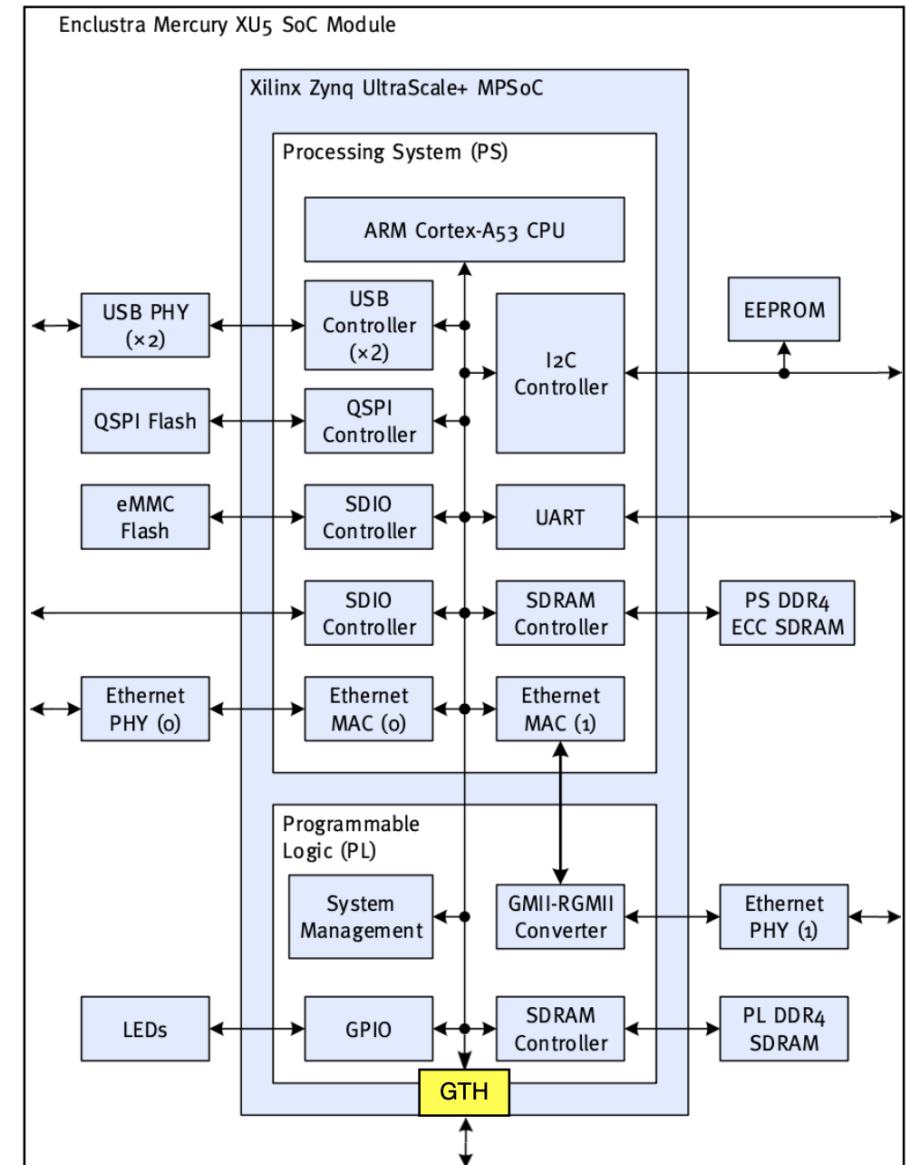
# Software Components

- Firmware for the big FPGA (Virtex Ultrascale+)



# Software Components

- Firmware for the MPSoC (Zynq)
- Embedded Linux (PetaLinux)
  - Run on ARM CPU on MPSoC

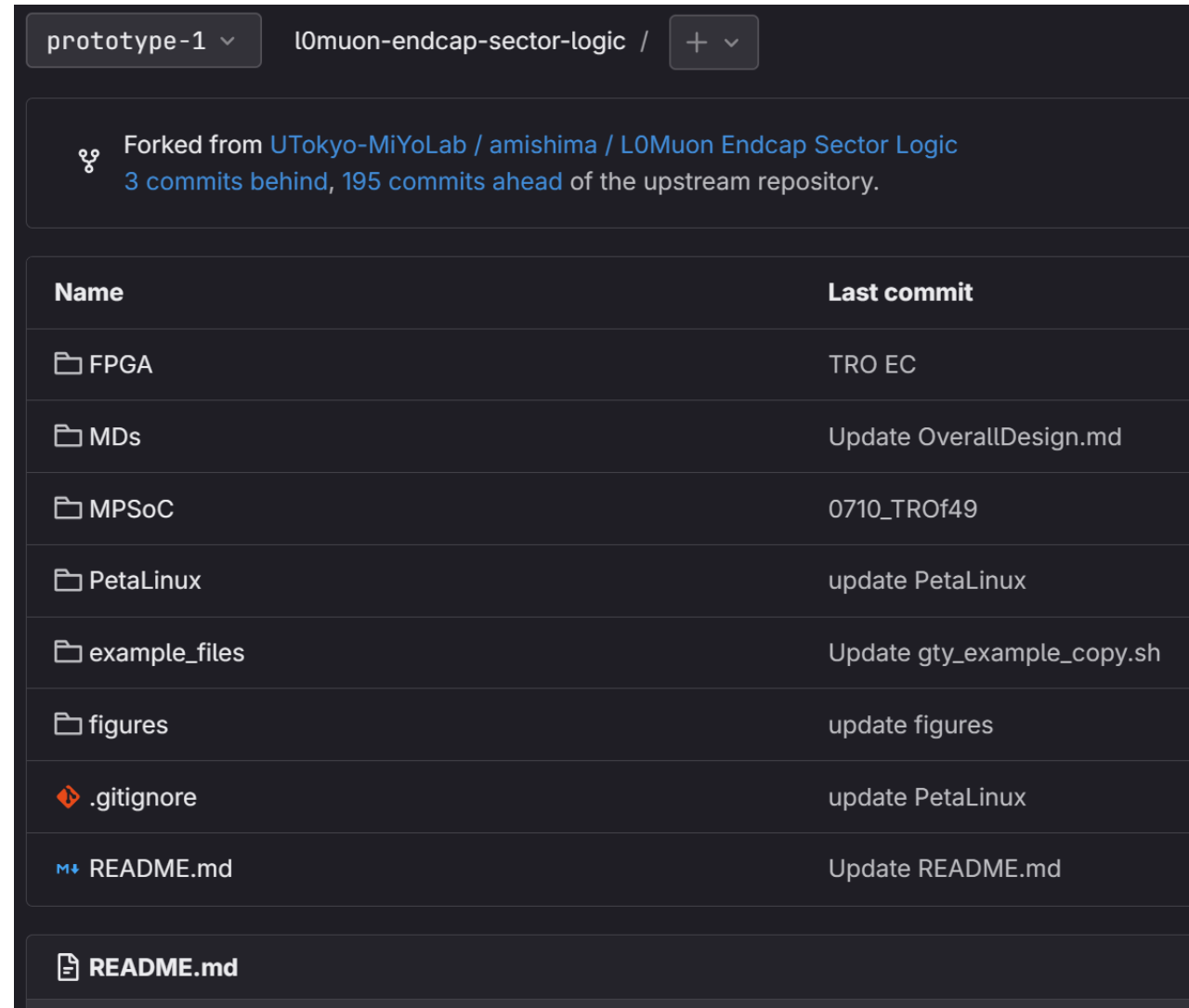




# SL Firmware on GitLab

# SL Firmware

- Originally, one project with a directory per component
- Components are:
  - Firmware for Virtex Ultrascale+ (FPGA)
  - Firmware for the MPSoC
  - Embedded Linux (PetaLinux) for the MPSoC
  - Documentations in MDs directory



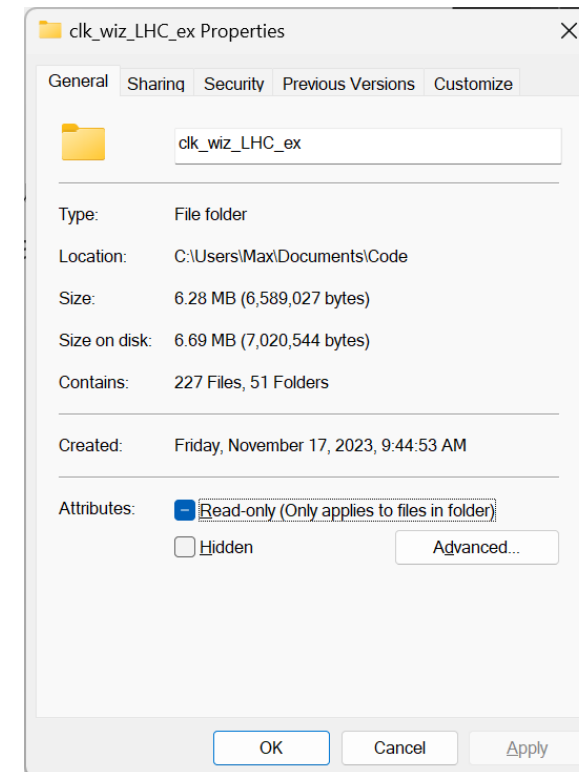
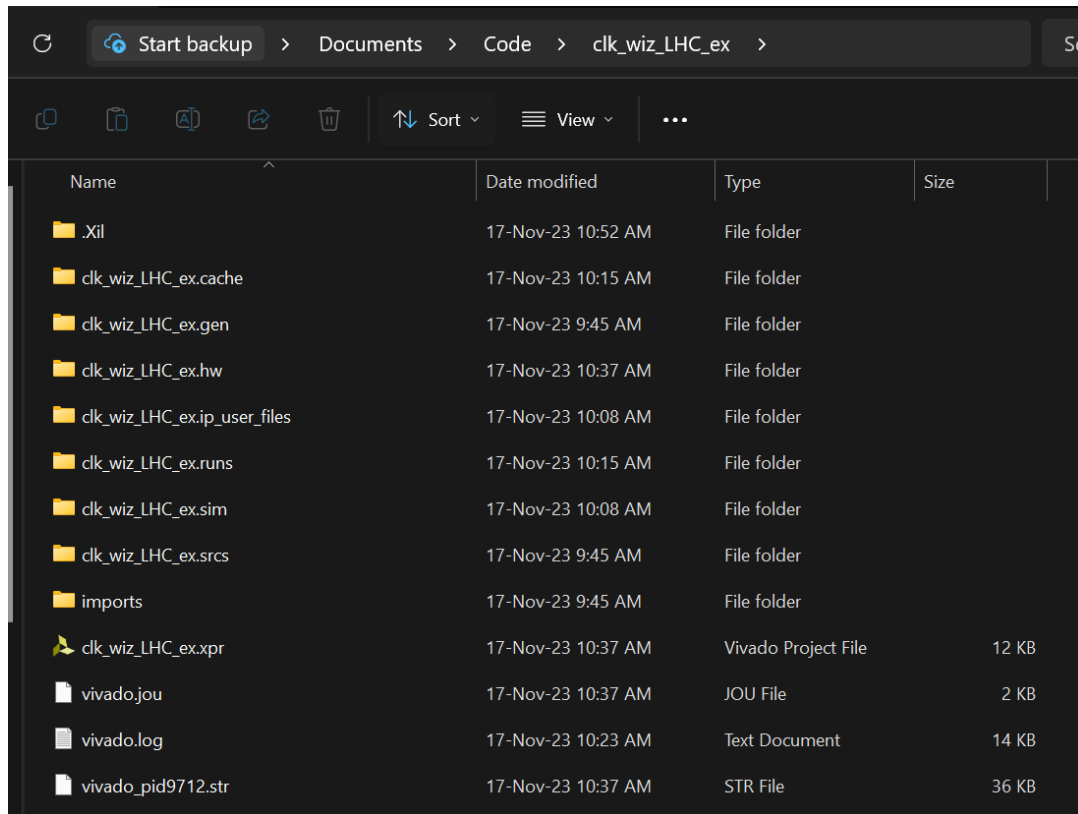
The screenshot shows a GitHub repository interface. At the top, there are dropdown menus for 'prototype-1' and 'l0muon-endcap-sector-logic'. Below this, it indicates the repository is forked from 'UTokyo-MiYoLab / amishima / L0Muon Endcap Sector Logic' and is '3 commits behind, 195 commits ahead of the upstream repository'. The main content is a table listing files and folders with their last commit information.

Name	Last commit
📁 FPGA	TRO EC
📁 MDs	Update OverallDesign.md
📁 MPSoC	0710_TROf49
📁 PetaLinux	update PetaLinux
📁 example_files	Update gty_example_copy.sh
📁 figures	update figures
📄 .gitignore	update PetaLinux
📄 README.md	Update README.md

At the bottom of the table, there is a separate entry for 'README.md' with a document icon.

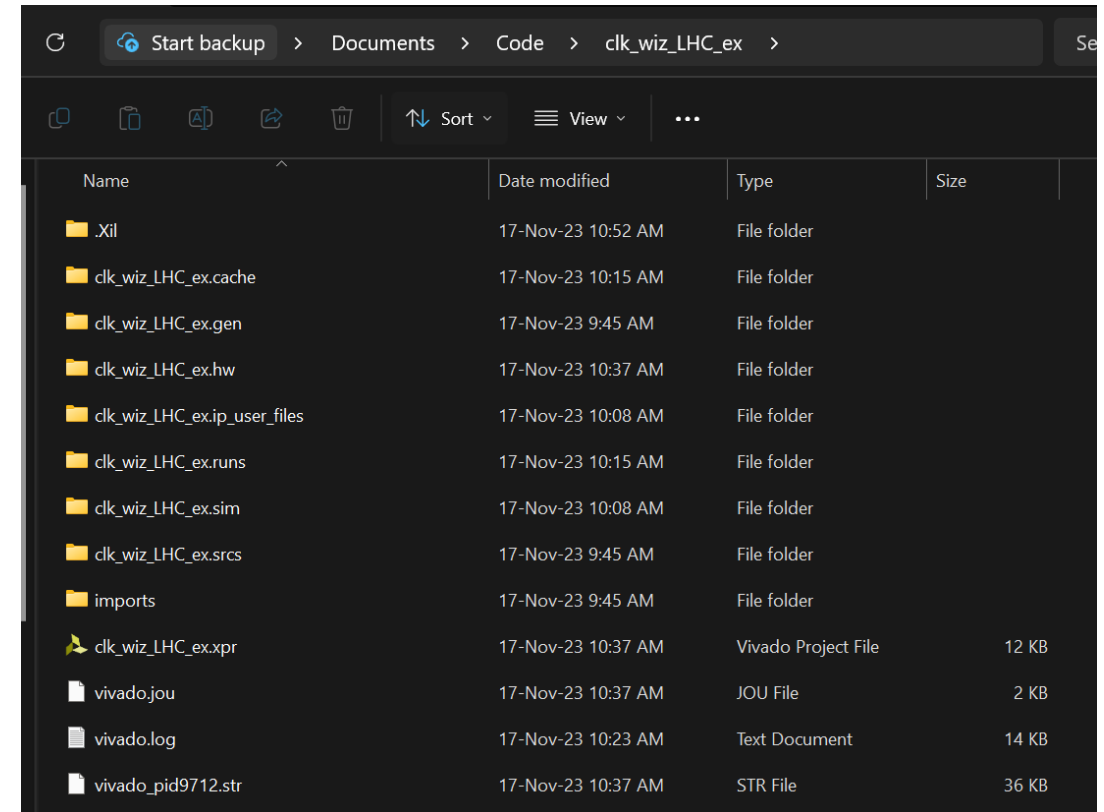
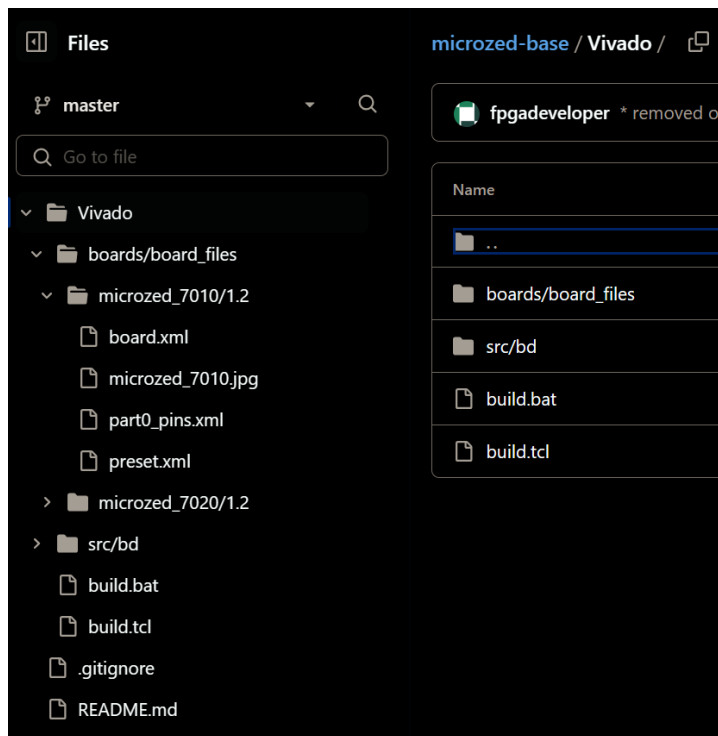
# Vivado Project on Git

- Putting Vivado Project on Git is not simple
  - Many temporary files are scattered in many directories
  - Many proprietary IP Core only provided as binary
    - Git was designed to track changes in plain text files



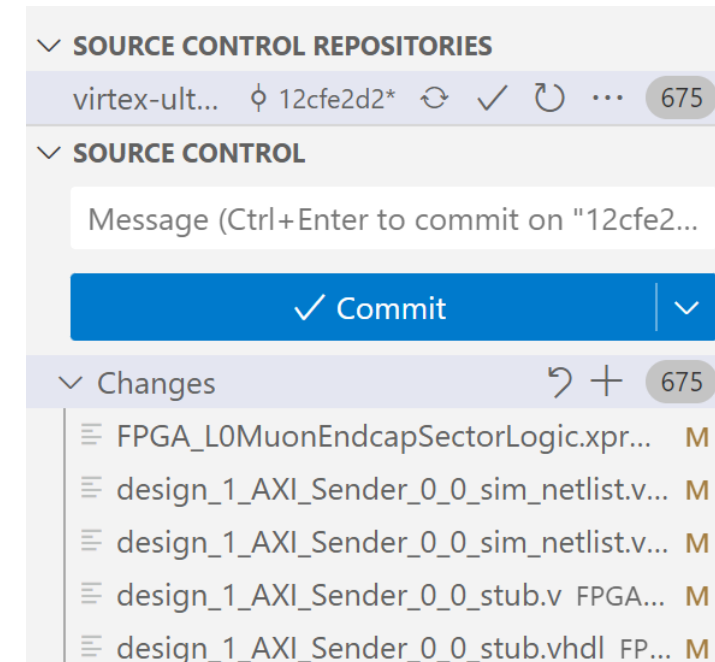
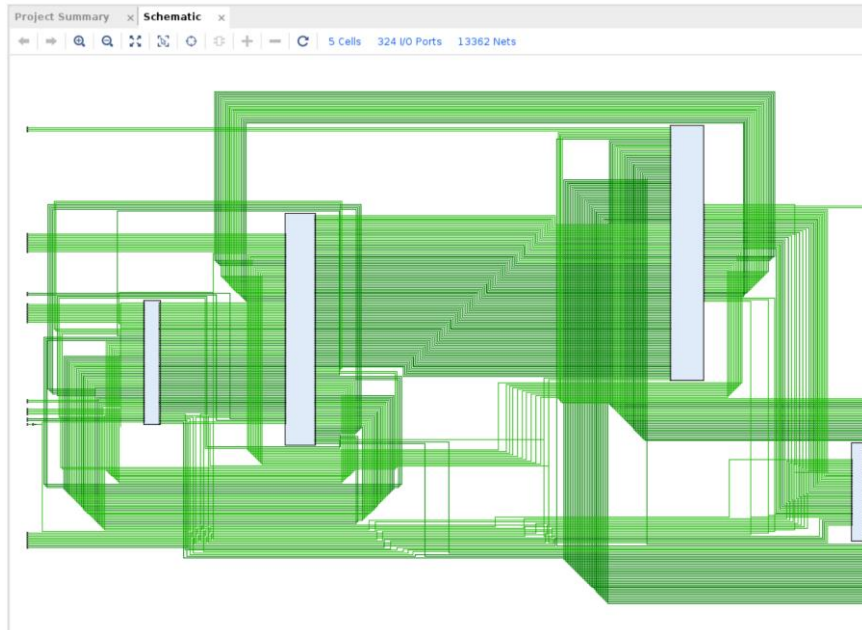
# Git-friendly vs. User-friendly

- Git-friendly: Use project-building script
  - + Minimum number of tracked files
  - + Clean commit history
  - - User need to pick what to push
- User-friendly: Track everything
  - + Minimum interventions from users
  - - Unnecessary changes in commits
  - - Very large repository



# Git-friendly vs. User-friendly

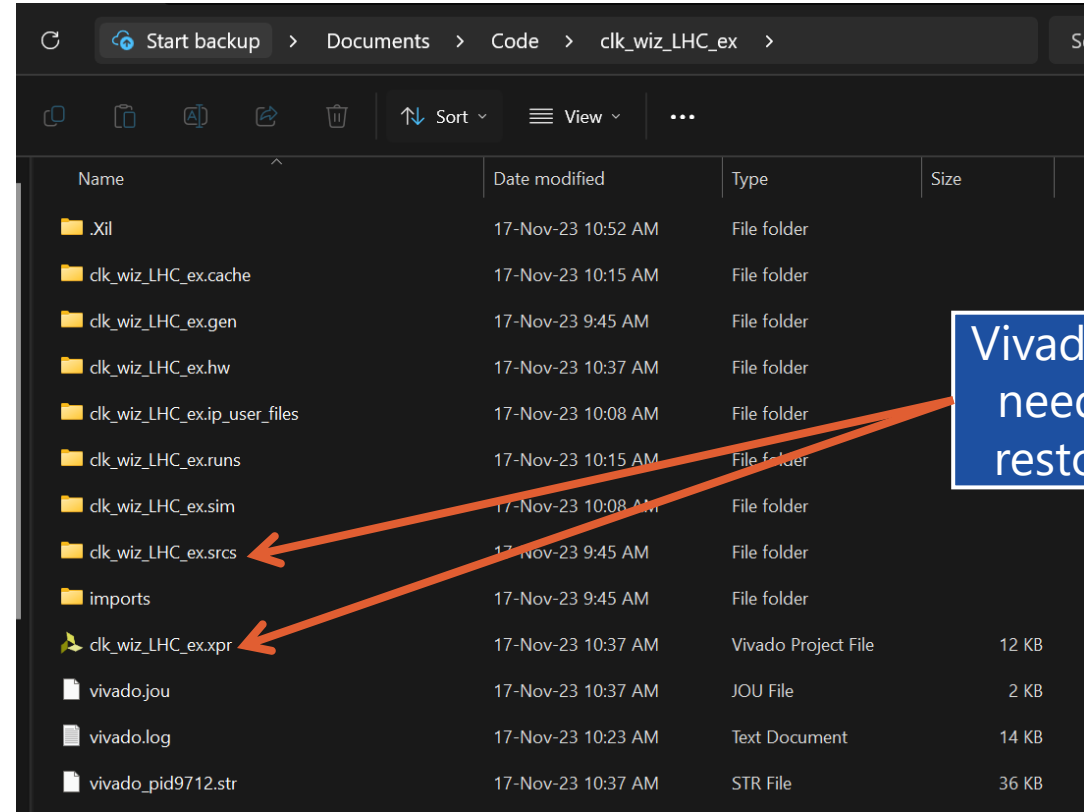
- Git-friendly: Many tools such as [Hog \(HDL on Git\)](#) can help
  - Opinion: Still complicate
- User-friendly: Use .gitignore to minimize tracking unneeded files
  - Still messy
  - For SL firmware:
    - Open schematic produces 675 "changes"
    - 5000+ for generating bitstream



# Good News for v2020.2

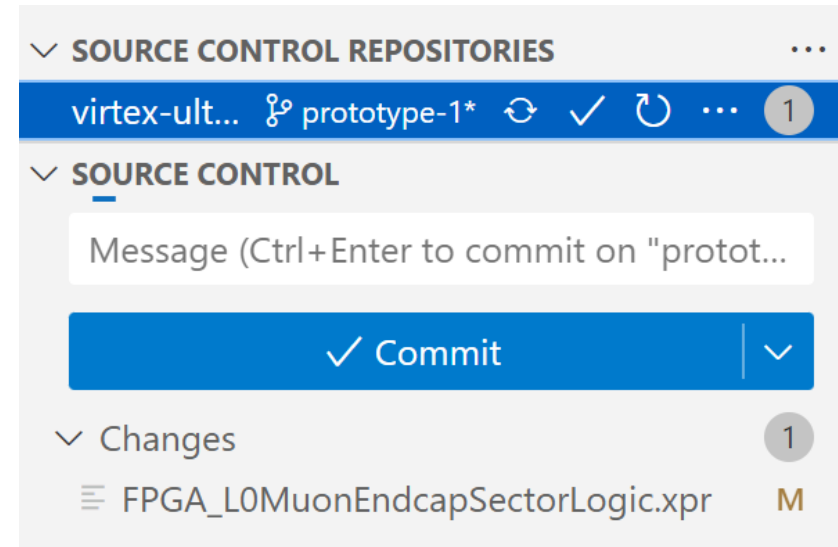
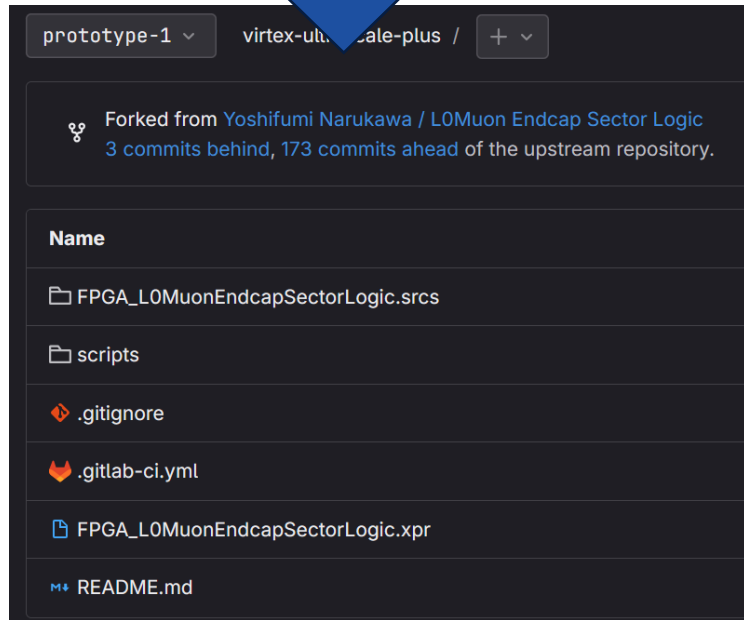
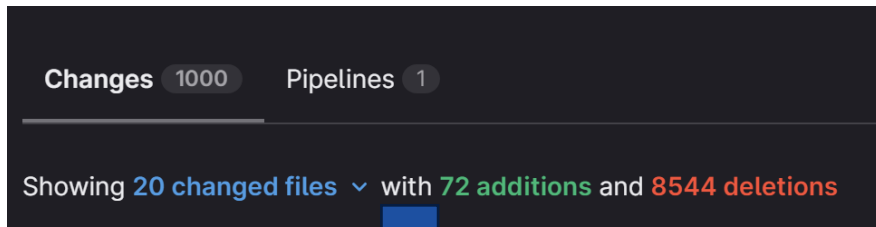
Revision Control Philosophy from Vivado 2020.2 Onwards [[Link](#)]

- In 2020.2, significant improvements are made to the Vivado project directory structure to improve the ability to interact with the revision control systems. [...] The project can be re-created by restoring the project.srcs directory and project.xpr file.



Vivado >2020.2 only need these two to restore the project

# New Directory Structure

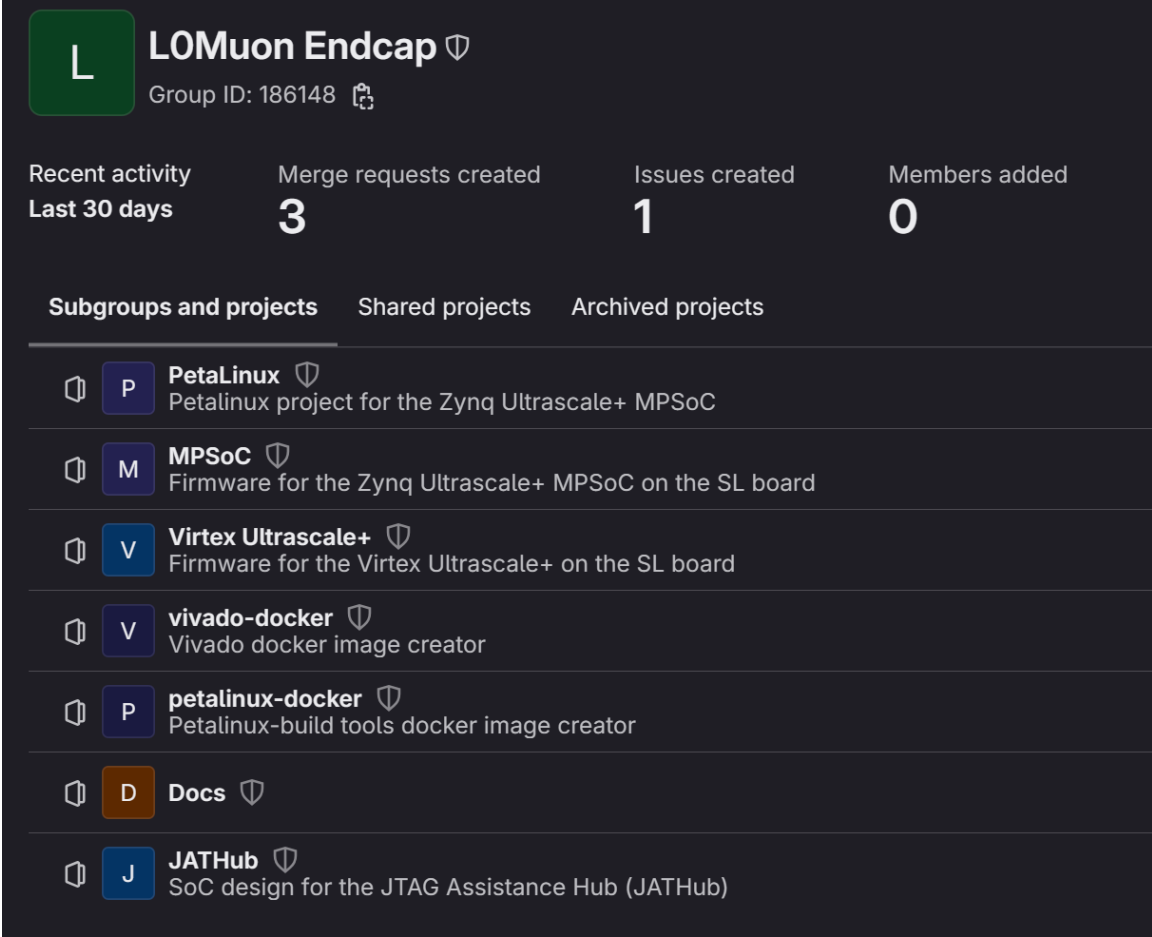


No more thousands of changes after simply opening the schematic

- Easy to put on GitLab
- User work in Vivado normally. Commit all tracked files and directories
- Not as clean as Hog Package, but hopefully much simpler for users

# New Project Structure for SL

- Separate each component to be its own project in GitLab
  - All under the same "group"



The screenshot displays the GitLab interface for the 'LOMuon Endcap' group. At the top, the group name 'LOMuon Endcap' is shown with a shield icon and 'Group ID: 186148'. Below this, a summary row shows 'Recent activity Last 30 days', 'Merge requests created 3', 'Issues created 1', and 'Members added 0'. A navigation bar includes 'Subgroups and projects' (selected), 'Shared projects', and 'Archived projects'. The main content area lists several projects, each with a folder icon, a letter in a colored box, the project name, a shield icon, and a description:

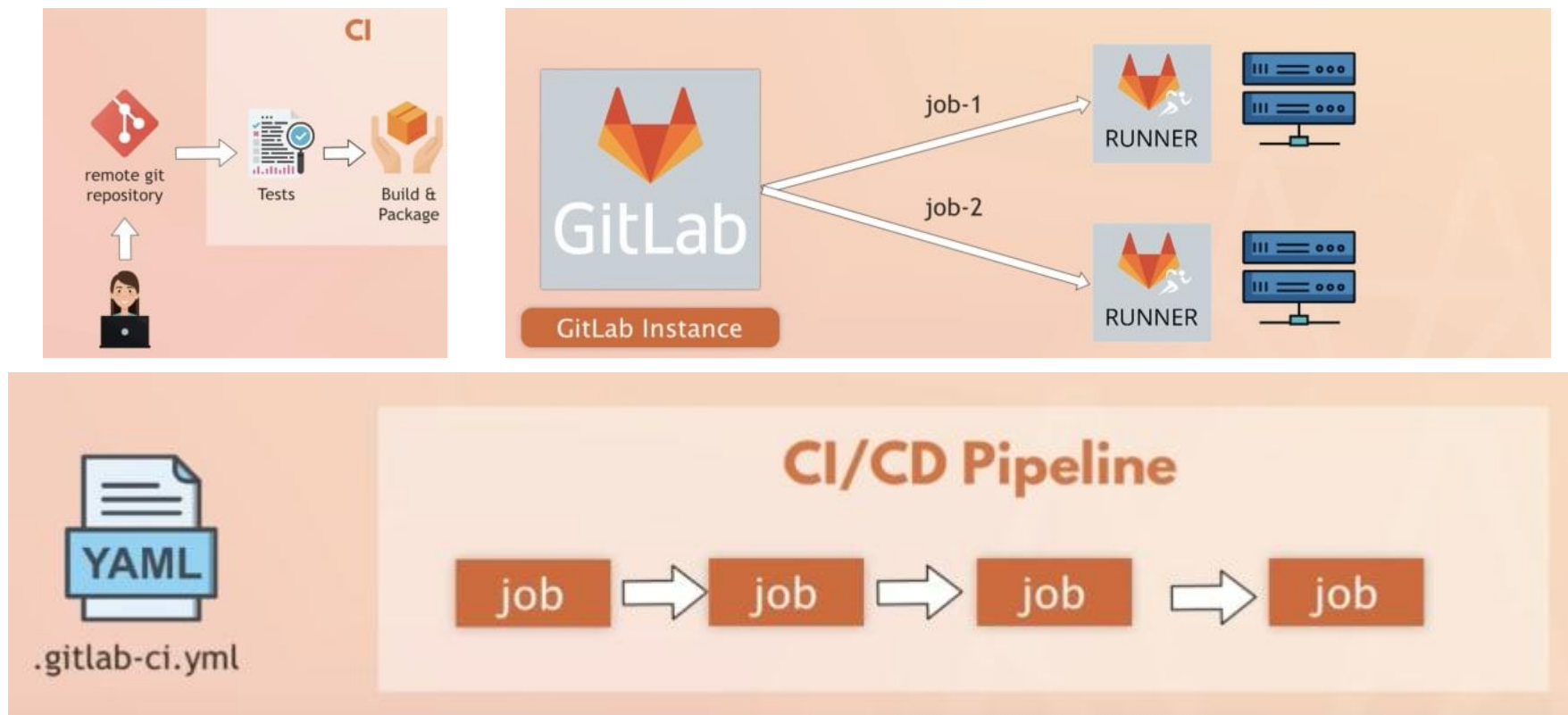
- P** **PetaLinux** Petalinux project for the Zynq Ultrascale+ MPSoC
- M** **MPSoC** Firmware for the Zynq Ultrascale+ MPSoC on the SL board
- V** **Virtex Ultrascale+** Firmware for the Virtex Ultrascale+ on the SL board
- V** **vivado-docker** Vivado docker image creator
- P** **petalinux-docker** Petalinux-build tools docker image creator
- D** **Docs**
- J** **JATHub** SoC design for the JTAG Assistance Hub (JATHub)



# GitLab-CI for SL Firmware

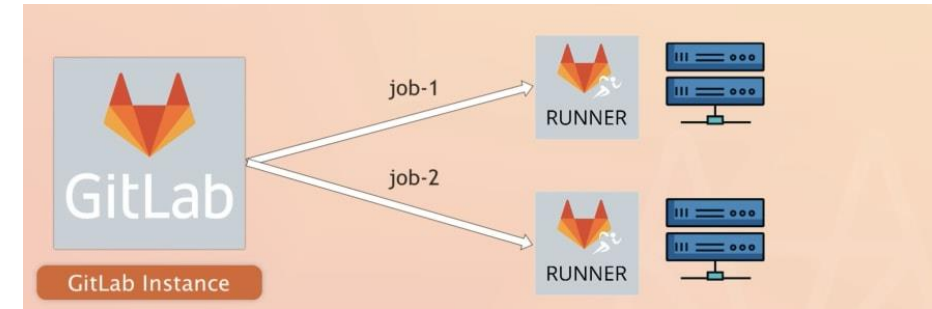
# GitLab CI/CD

- Automate check (simulation) and build (synthesize, implement, generate bitstream)
- Triggered when new commits are pushed and run on machines with `gitlab-runner` installed
- A YAML file (`.gitlab-ci.yml`) tell GitLab what to do (defines jobs)
  - A lot easier with each firmware on its own project



# GitLab Runner

- Software that run the jobs
- Compatible with KEK and UTokyo firewall



**Runners**

All 222 Group 5 Project 0

Search or filter results... Created date

Online 29 Offline 181 Stale 183 Upgrade available 0 Upgrade recommended 3

Status	Runner	Owner
Online Idle	#33746 (R21MC-UFp) Instance Version 16.4.2 · default-runner-64c64d5d74-v9tpz Last contact: 15 minutes ago · 188.185.15.160 · 109 · Created 1 day ago docker no-runner-9	Administrator
Online Idle	#33741 (zBXDPzf6J) Instance Version 16.4.2 · default-runner-64c64d5d74-k8wdx Last contact: 15 minutes ago · 188.185.22.120 · 332 · Created 2 days ago docker no-runner-9	Administrator
Online Running	#33740 (Rgc6svwQV) Instance Version 16.4.2 · default-runner-64c64d5d74-x89c6	Administrator

Shared Runners from CERN

**Runners**

All 222 Group 5 Project 0

Search or filter results... Created date

Online 5 Offline 0 Stale 0 Upgrade available 0 Upgrade recommended 2

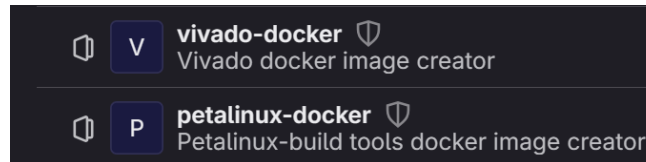
Status	Runner	Owner
Online Idle	#33433 (ukhRXchQC) Group Version 16.5.0 · lncelec09-docker Last contact: 58 minutes ago · 2001:200:180:1105:e63d:1aff:fe4f:344c · 12 Created 3 weeks ago by vivado_2020.2	LOMuon Endcap
Online Idle	#33350 (8ELxSo-g4) Group Version 16.2.0 · lncelec13-docker Last contact: 13 minutes ago · 2001:200:180:1105:e63d:1aff:fe4f:b5e2 · 12 Created 1 month ago vivado_2020.2	LOMuon Endcap
Online Paused Idle	#33088 (e_-xKbWo6) Group Version 16.5.0 · KEK-Minami-Ubuntu-Docker Last contact: 13 minutes ago · 130.87.243.191 · 139 Created 2 months ago vivado_2020.2	LOMuon Endcap

Private Runners at KEK and UTokyo

[GitLab CI/CD for Beginners \[FREE Course\] - DEV Community](#)

# Executors and Docker

- Many of “Executors” for GitLab Runner. We use:
  - Shell: Like running in a terminal
  - Docker: Run in a Docker container
- Prefer Docker executor for future compatibility
  - Experience: Upgrade Ubuntu 20.04 LTS to 22.04 LTS break petalinux build
  - Scripts to build them are also on GitLab



	<b>vivado-docker image</b>	<b>petalinux-docker image</b>
Software	<ul style="list-style-type: none"><li>• Ubuntu 18.04</li><li>• Vivado 2020.2+dependencies</li></ul>	<ul style="list-style-type: none"><li>• Ubuntu 18.04</li><li>• Petalinux 2020.2+dependencies</li></ul>
Size	~50 GB	~11 GB
License	Need	No need
Target Runner	Private machines (KEK/UTokyo)	CERN Share runners

# .gitlab-ci.yml

- Example from FPGA firmware (simplified)

```
image:
  name: gitlab-registry.cern.ch/l0muon-endcap/vivado-docker:2020.2

stages: # List of stages for jobs, and their order of execution
  - check_syntax
  - sim
  - synth
  - impl
  - gen_bitstream


check_syntax-job:
  ...
  ...

gen_bitstream-job:
  stage: gen_bitstream
  script:
    - echo "Running gen_bitstream"
    - vivado -mode batch -source scripts/gen_bitstream.tcl -tclargs ${PROJECT_NAME}.xpr
  artifacts:
    paths:
      - ${PROJECT_NAME}.runs/impl_1/
    exclude:
      - ${PROJECT_NAME}.runs/impl_1/*.dcp
```

# .gitlab-ci.yml

- "image" sector specify the docker image to use

```
image:  
  name: gitlab-registry.cern.ch/l0muon-endcap/vivado-docker:2020.2
```



```
Running with gitlab-runner 16.2.0 (e15da)  
2 on utokyo docker 8ELxSo-g, system_45657d31d342  
3 Resolving secrets 00:00  
5 Preparing the "docker" executor 59:41  
6 Using Docker executor with image gitlab-registry.cern.ch/l0muon-endcap/vivado-docker:2020.2 ...  
7 Using helper image: registry.gitlab.com/gitlab-org/gitlab-runner/gitlab-runner-helper:x86_64-782e15da  
8 Pulling docker image registry.gitlab.com/gitlab-org/gitlab-runner/gitlab-runner-helper:x86_64-782e15da ...  
9 Using docker image sha256:e25833d65f8929de622e5f7587e9fd7975524c0cefc573f1b605049b18b2c97a for  
registry.gitlab.com/gitlab-org/gitlab-runner/gitlab-runner-helper:x86_64-782e15da with digest registry.gitlab.com/gitlab-  
org/gitlab-runner/gitlab-runner-  
helper@sha256:a5fba688be690f75d7c0e5c6b668a6a8f50b041e83981f275d55b0ef1f0ec727 ...  
10 Authenticating with credentials from job payload (GitLab Registry)  
11 Pulling docker image gitlab-registry.cern.ch/l0muon-endcap/vivado-docker:2020.2 ...  
12 Using docker image sha256:513177a800eb544d4075e1db893011ac33769b7650b35c9c421adab76583521c for gitlab-  
registry.cern.ch/l0muon-endcap/vivado-docker:2020.2 with digest gitlab-registry.cern.ch/l0muon-endcap/vivado-  
docker@sha256:80aab9092561035fcfd4b1b5bcd9a2aa2081f82087c3cb4bda312d16cba577c ...  
14 Preparing environment 00:19  
15 Running on runner-8elxso-g-project-165910-concurrent-0 via lhcelec13...
```

# .gitlab-ci.yml

- "stage" section list the jobs in order

```
image:  
  name: gitlab-registry.cern.ch/l0muon-endcap/vivado-docker:2020.2
```

```
stages: # List of stages for jobs, and their order of execution
```

- check\_syntax
- sim
- synth
- impl
- gen\_bitstream

```
check_syntax-job:  
...
```

```
gen_bitstream-job:  
  stage: gen_bitstream
```

```
  script:  
    - echo "Running gen_bitstream"  
    - vivado -mode batch -source scripts/gen_bitstream.tcl -tclargs ${PROJECT_NAME}.xpr  
  artifacts:  
    paths:  
      - ${PROJECT_NAME}.runs/impl_1/  
    exclude:  
      - ${PROJECT_NAME}.runs/impl_1/*.dcp
```

L0Muon Endcap > Virtex Ultrascale+ > Pipelines > #6419239

## First fix

passed Chaowaroj Wanotayaroj created pipeline for commit [a5acb4e0](#) finished 2 weeks ago

For [newTrackSel-TimingFix](#)

latest 5 Jobs 0 Tests 1201 minutes 41 seconds, queued for 3 seconds

Pipeline Needs Jobs 5 Tests 0

check\_syntax

sim

synth

impl

gen\_bitstream

check\_syntax-job

sim-job

synth-job

impl-job

gen\_bitstream-job

# .gitlab-ci.yml

- The "script" section list commands to run

```
check_syntax-job:
  stage: check_syntax
  script:
    - echo "Checking syntax"
    - vivado -mode batch -source scripts/check_syntax.tcl -tclargs ${PROJECT_NAME}.xpr
```



```
14 Getting source from Git repository00:14
15 Fetching changes with git depth set to 20...
16 Reinitialized existing Git repository in /builds/I0muon-endcap/virtex-ultrascale-plus/.git/
17 Checking out a5acb4e0 as detached HEAD (ref is newTrackSel-TimingFix)...
...
27 Executing "step_script" stage of the job script06:52
28 Using docker image sha256:513177a800eb544d4075e1db893011ac33769b7650b35c9c421adab76583521c for gitlab-
registry.cern.ch/I0muon-endcap/vivado-docker:2020.2 with digest gitlab-registry.cern.ch/I0muon-endcap/vivado-
docker@sha256:80aab9092561035fcfd4b1b5bcd9a2aa2081f82087c3cb4bda312d16cba577c ...
29 $ source /tools/Xilinx/Vivado/2020.2/settings64.sh
30 $ source $LICENSE
31 $ echo "Checking syntax"
32 Checking syntax
33 $ vivado -mode batch -source scripts/check_syntax.tcl -tclargs ${PROJECT_NAME}.xpr
```



# .gitlab-ci.yml

- The "artifacts" section tells what to keep (or not)

```
artifacts:  
  paths:  
    - ${PROJECT_NAME}.runs/impl_1/  
  exclude:  
    - ${PROJECT_NAME}.runs/impl_1/*.dcp
```



```
Uploading artifacts for successful job00:16  
1218Uploading artifacts...  
1219FPGA_L0MuonEndcapSectorLogic.runs/impl_1/: found 65 matching artifact files and  
directories  
1220FPGA_L0MuonEndcapSectorLogic.runs/impl_1/*.dcp: excluded 4 files  
1221Uploading artifacts as "archive" to coordinator... 201 Created id=33590155  
responseStatus=201 Created token=64_y5VsM  
1223Cleaning up project directory and file based variables00:01  
1225Job succeeded
```



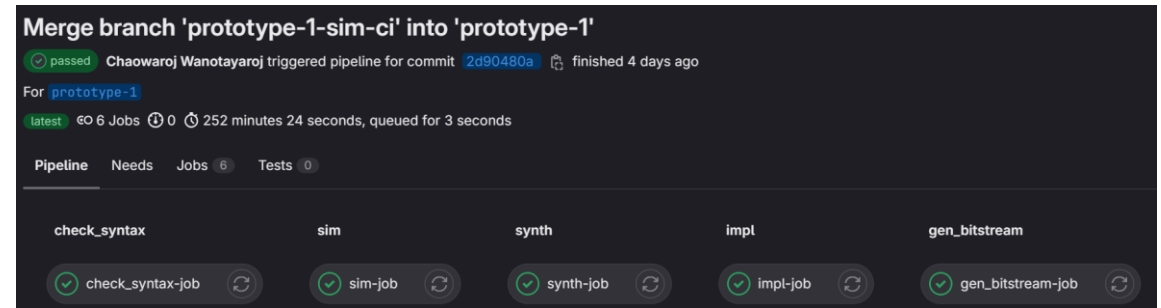
passed Job #33590155 in pipeline #6419239 for a5acb4e0 from newTrackSel-TimingFix

Artifacts

Name
FPGA_L0MuonEndcapSectorLogic.runs

# Current CI Pipeline Status

- Pipeline
  - Virtex Ultrascale+
  - MPSoC
  - Petalinux
- Vivado & Petalinux docker image are ready to be used
  - [gitlab-registry.cern.ch/l0muon-endcap/vivado-docker](https://gitlab-registry.cern.ch/l0muon-endcap/vivado-docker)
    - Need license for your device
  - [gitlab-registry.cern.ch/l0muon-endcap/petalinux-docker](https://gitlab-registry.cern.ch/l0muon-endcap/petalinux-docker)
- Associate documentations: static website auto-created by GitLab-CI (see [backup](#))



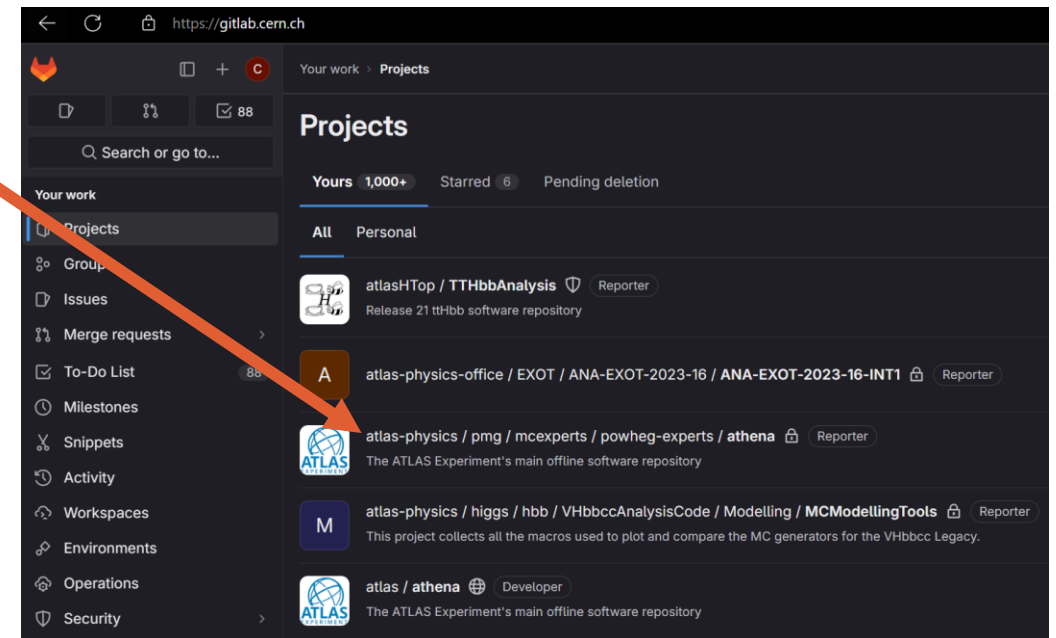
# Summary

- Git is an essential tool for collaborative software development
- GitLab-CI can automate building and testing
- FPGA design workflow using Vivado is not very git-friendly, but in v2020.2 a significant change has been made to address the issue
- We restructure the GitLab projects to achieve:
  - Large reduction of the number of tracking files
  - A very simple Vivado workflow for users
- Build pipelines are tested and working well
- Compile the build environment (OS, libraries, tools, etc.) into docker images for future compatibility

# Backup

# Introduction - Git vs. GitHub

- A service such as GitHub or GitLab let you host a Git repository
- And offer other features on top of it
  - Bug tracking
  - Automate building (compiling, etc.)
- CERN host their own GitLab instance
  - Can't do that with GitHub
- GitLab projects can be organized in a group
  - E.g. <Group1>/<Subgroup1>/<Project Name>



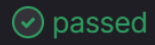
# GitLab CI/CD

- From GitLab page on CI/CD:
  - CI/CD falls under DevOps (the joining of development and operations teams) and combines the practices of [continuous integration](#) and [continuous delivery](#). CI/CD automates much or all of the manual human intervention traditionally needed to get new code [from a commit into production, encompassing the build, test](#) (including integration tests, unit tests, and regression tests), and deploy phases, as well as infrastructure provisioning. With a CI/CD pipeline, development teams can make changes to code that are then automatically tested and pushed out for delivery and deployment.
- CERN IT offers many DevOps solutions such as Jenkins which is more advance than GitLab CI/CD
- We only needs an automate build and test systems
  - GitLab CI/CD can do that and nicely integrated into GitLab web interface
- The actual build and test jobs are done on machine with [gitlab-runner](#) installed
  - CERN project a shared pool for all project on CERN GitLab
  - gitlab-runner software can be installed and register as private runner for a project or a group
- GitLab looks for [.gitlab-ci.yml](#) file in the repository for CI directive

# PetaLinux Builder Tools



- As a bonus for having a docker image for building PetaLinux, it can run on Shared Runners pool from CERN
  - Compatible with the new Kubernetes clusters

```
[OK]Running with gitlab-runner 15.10.1 (dcfb4b66)[0;m
[OK] on runners-k8s-default-runners-575cd88bc4-rk99r PVQ-GR3u, system ID: r_jWkAVo4T5Rss[0;m
[OK] feature flags: FF_KUBERNETES_HONOR_ENTRYPOINT:true, FF_USE_ADVANCED_POD_SPEC_CONFIGURATION:true[0;m
section_start:1693440766:resolve_secrets
[OK][OK[36;1mResolving secrets[0;m[0;m
section_end:1693440766:resolve_secrets
[OK]section_start:1693440766:prepare_executor
[OK][OK[36;1mPreparing the "kubernetes" executor[0;m[0;m
[OK]Using Kubernetes namespace: gitlab[0;m
[OK]Using Kubernetes executor with image gitlab-registry.cern.ch/l0muon-endcap/petalinux-builder:2020.2 ...[0;m
[OK]Using attach strategy to execute scripts...[0;m
```

Status	Job	Stage	Name
 passed	#32111997 <a href="#">Prototype1_1.0</a> ↻ 23856f1a	petalinux-build	petalinux-build-job















# New Project Structure

- Separate each firmware to be its own project in GitLab
  - All under the same "group"
- Allow individual CI config (.gitlab-ci.yml)
- Git Tagging to synchronize compatible releases
- Any project that is sitting under personal namespace can be forked over to group namespace without losing history
  - Easy migration of any project anywhere on CERN GitLab instance

**L** **LOMuon Endcap**   
Group ID: 186148 

Recent activity	Merge requests created	Issues created	Members added
Last 30 days	<b>3</b>	<b>1</b>	<b>0</b>

**Subgroups and projects**   Shared projects   Archived projects

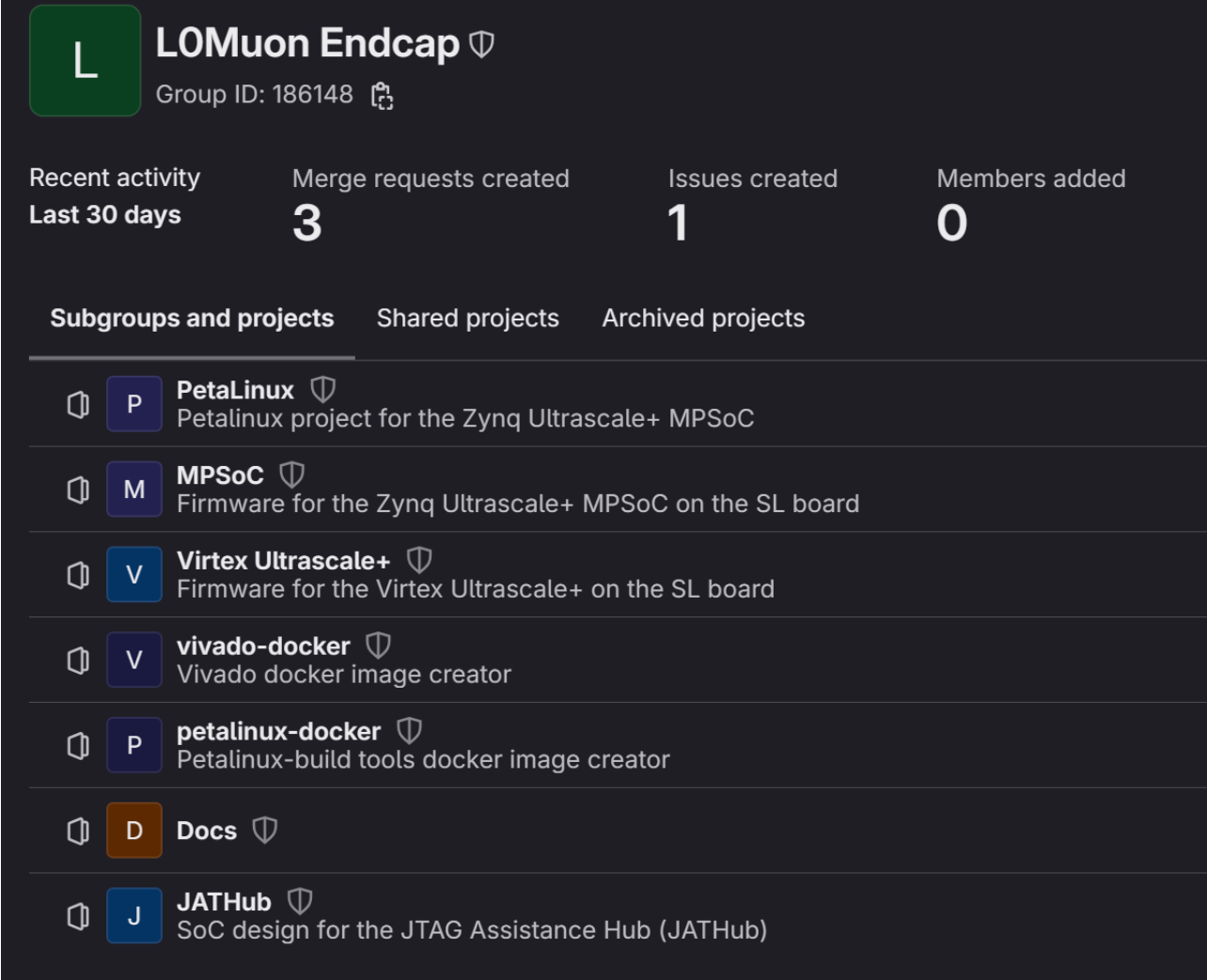
-  **P** **PetaLinux**   
Petalinux project for the Zynq Ultrascale+ MPSoC
-  **M** **MPSoC**   
Firmware for the Zynq Ultrascale+ MPSoC on the SL board
-  **V** **Virtex Ultrascale+**   
Firmware for the Virtex Ultrascale+ on the SL board
-  **V** **vivado-docker**   
Vivado docker image creator
-  **P** **petalinux-docker**   
Petalinux-build tools docker image creator
-  **D** **Docs** 
-  **J** **JATHub**   
SoC design for the JTAG Assistance Hub (JATHub)



# New Project Structure

## Current Projects:

- Docs
- Virtex Ultrascale+
  - Large FPGA on the SL board
- MPSoC
  - Zynq FPGA on the mezzanine board
- PetaLinux
  - MPSoC's Linux distro image builder
- vivado-docker
  - Docker image builder for Vivado
- petalinux-docker
  - Docker image builder for petalinux
- JATHub



The screenshot shows the GitHub organization page for "LOMuon Endcap". At the top, there is a green profile picture with the letter 'L', the organization name "LOMuon Endcap" with a shield icon, and the "Group ID: 186148" with a lock icon. Below this, there are four statistics: "Recent activity Last 30 days", "Merge requests created 3", "Issues created 1", and "Members added 0". Underneath, there are three tabs: "Subgroups and projects", "Shared projects", and "Archived projects". The "Subgroups and projects" tab is active, showing a list of projects with their respective icons and descriptions:

- PetaLinux**: Petalinux project for the Zynq Ultrascale+ MPSoC
- MPSoC**: Firmware for the Zynq Ultrascale+ MPSoC on the SL board
- Virtex Ultrascale+**: Firmware for the Virtex Ultrascale+ on the SL board
- vivado-docker**: Vivado docker image creator
- petalinux-docker**: Petalinux-build tools docker image creator
- Docs**
- JATHub**: SoC design for the JTAG Assistance Hub (JATHub)

# GitLab-CI

- Setup automatic pipeline with GitLab-CI
  - Automate check & build (synthesize, implement, generate bitstream)
- GitLab-CI will trigger a pipeline which will be run using machines at KEK and UTokyo
  - To our surprise, neither KEK nor UTokyo firewall is an issue for GitLab Runner
- GitLab Runner can be installed as many type of “Executors”
  - We use Shell Executor and Docker Executor
    - Shell Executor: simple, like running manually through terminal
    - Docker Executor: clean, static environment
- CERN has a pool or shared runners we can use
  - Used to be Docker executor, recently begin migrating to Kubernetes
  - Limitations:
    - Recommend not going over 10GB for the image
    - Vivado licenses we have are fixed per hardware and cannot be easily apply to the shared runners

# Running Pipelines

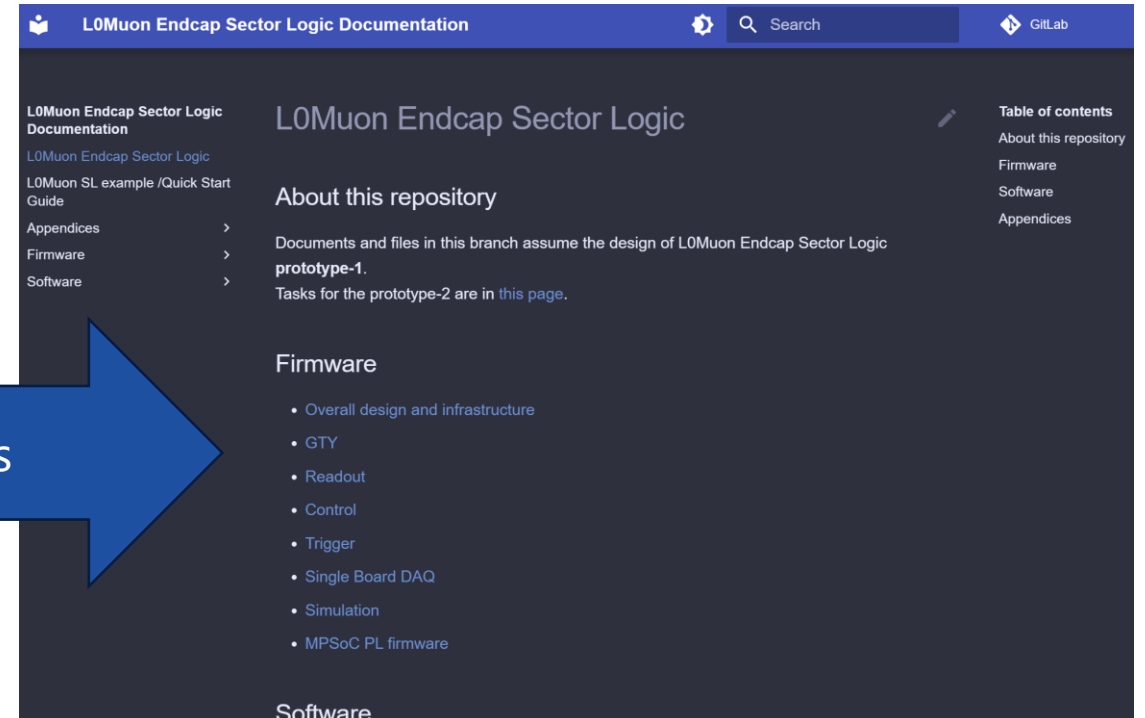
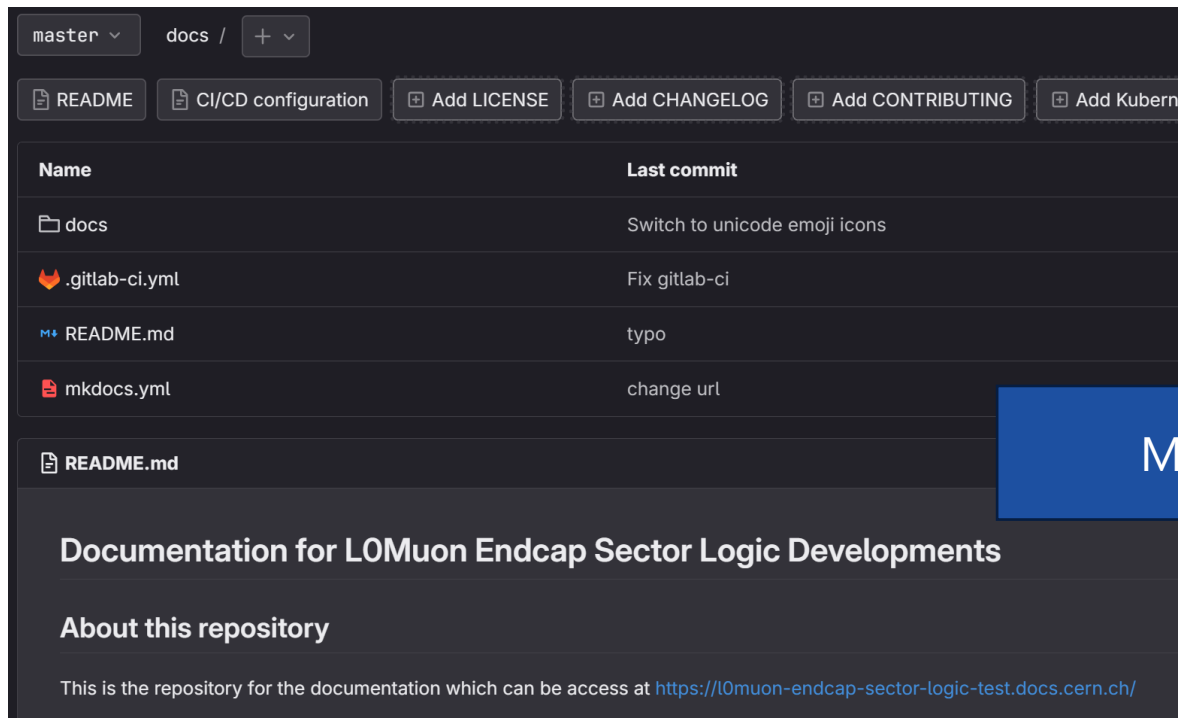
- Original idea:
  - Use shell executor. Keep it simple
  - Self-host runner (KEK and UTokyo)
- Given the timeline of Phase-II/Run4/5, it is important to ensure future compatibility
  - Already run into a problem upgrading from Ubuntu 20.04 LTS to 22.04 LTS. Petalinux 2020.2 can no longer run!?
- Switch to docker container
  - Clean and static environment for pipeline
  - Ensure future compatibility
  - Side benefits
    - Can run on CERN shared runner when license is not an issue (Petalinux). Tested on the new Kubernetes cluster
    - Provide a simple way to run Vivado on MacOS (only need Docker)

# Running Pipelines

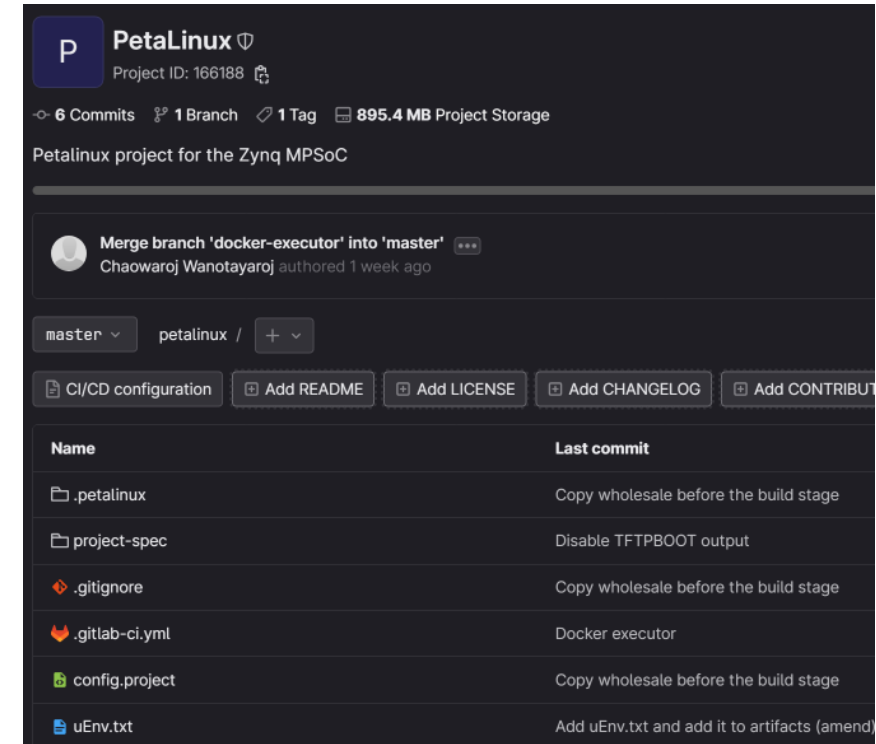
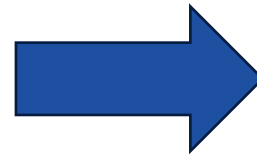
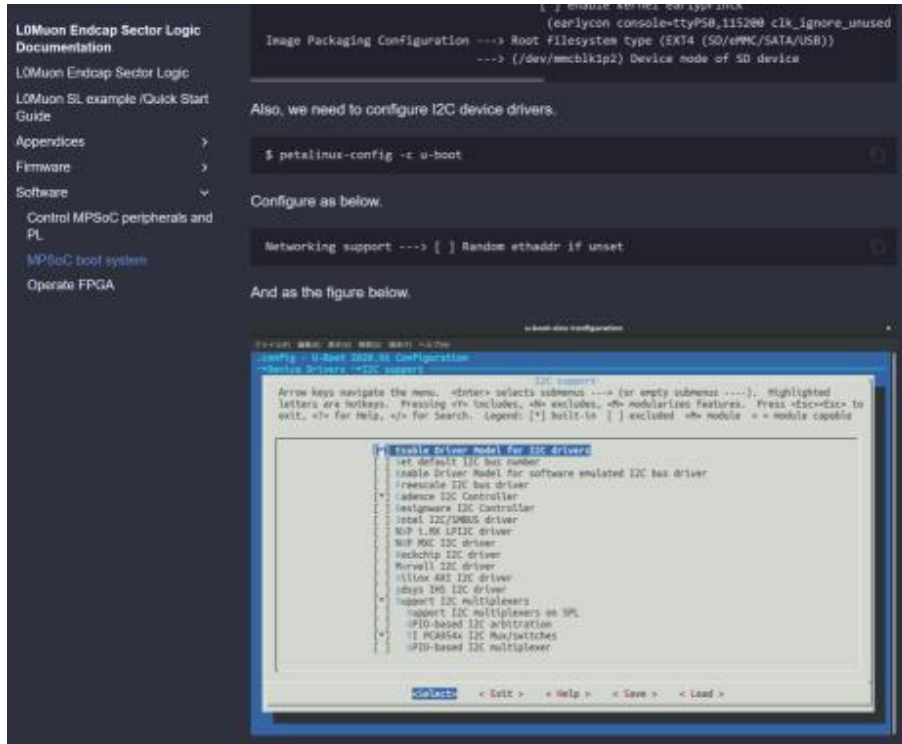
- Now both Petalinux and firmware (Virtex&Zynq) generations run on docker containers
- Petalinux: ~11GB image. No license needed. Can run on CERN share runners
- Vivado: ~50GB image. Need license. Most likely need dedicate machines
  - Due to the size, may want to change pull policy (always -> if-not-present)
  - Tested at both KEK and Utokyo
  - Runners available to all projects in l0muon group
- Dockerfiles and scripts to build the images are also on GitLab
  - But the image building itself was not intend to be used with CI
    - Can be done in theory with Kaniko, but downloading the installers are a pain and the images are not expected to be rebuilt many times

# Documentations

- Move all Markdown files (\*.md) to its own project
  - See details: [Create a MkDocs website \(like this one\) - ABP Computing @ CERN](#)
- Static documentation webpages auto-generated using MKDocs
  - <https://l0muon-endcap-sector-logic-test.docs.cern.ch/>



# PetaLinux on GitLab



- Before: A list of instruction on how to build an image with a few files scattered around the big project and GUI that need manual input
- After: A project on GitLab one can clone and build
  - Can be automated with CI