



HSE
Radiation Protection

Towards Modern Heterogenous Code Development for SoC : Application to CROME System



Clyde Laforge, Hamza Boukabache, CROME Team

3 October 2023

About me

- Electronic Engineer
- Joined CERN as Fellow in July 2022
- Working in HSE-RP-IL
 - CERN Radiation Monitoring Electronics (CROME)
- Previous talks at SoC Interest Group Meetings:
 - [Paying off technical debt of SoC code-bases through standards and good practices](#)
 - [Containerization as a means of extending the lifetime of HDL development tools](#)
- Don't hesitate to ask questions: clyde.laforge@cern.ch

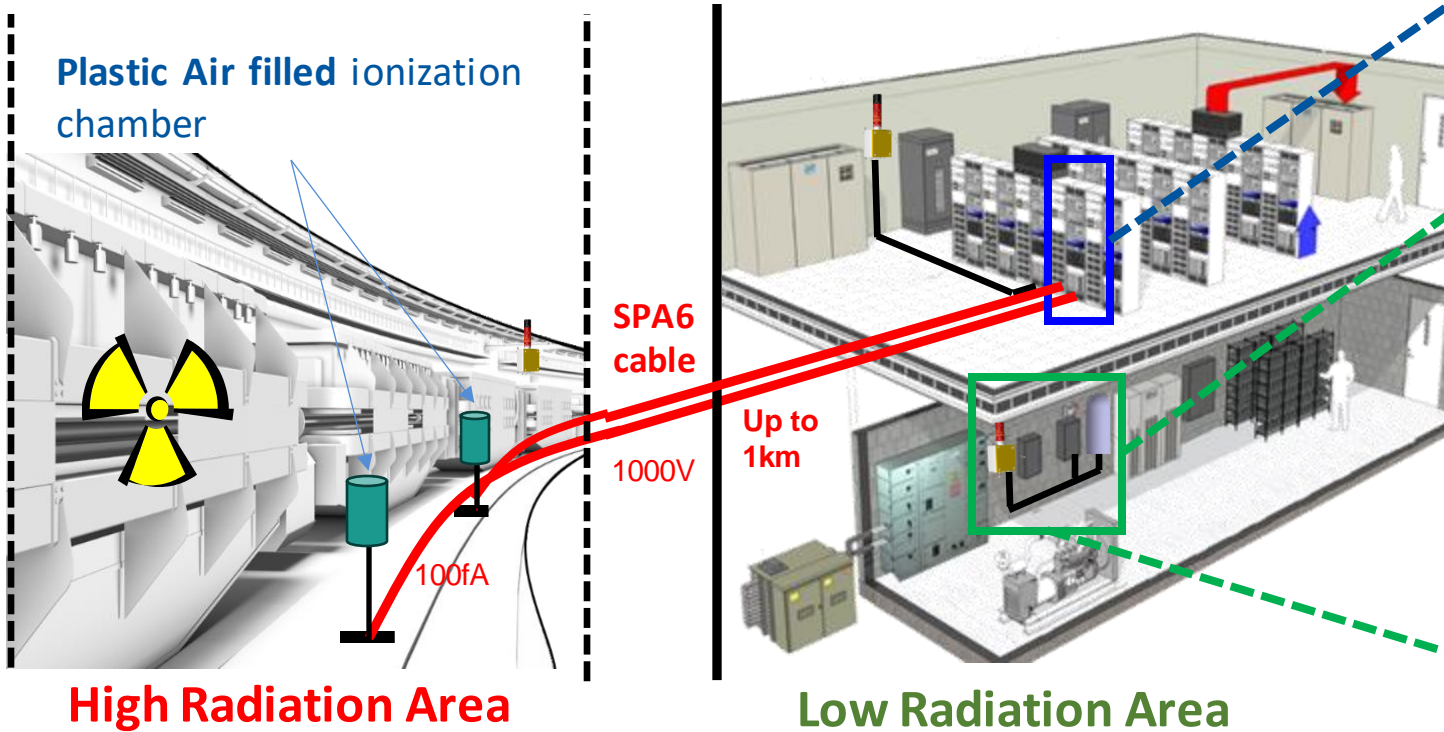


CROME Project

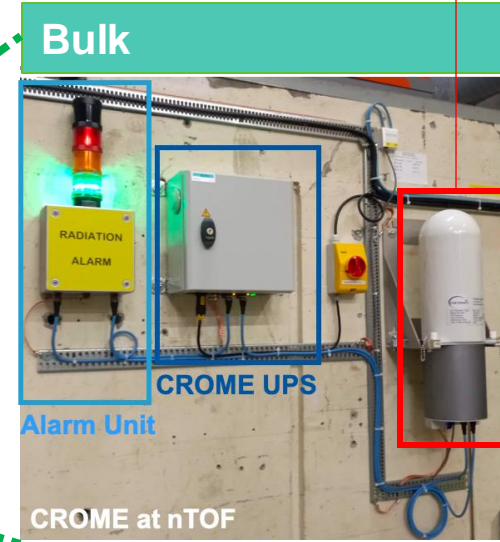
CERN Radiation Monitoring Electronics (CROME)

Two configurations :

Conceptual view of CROME at CERN



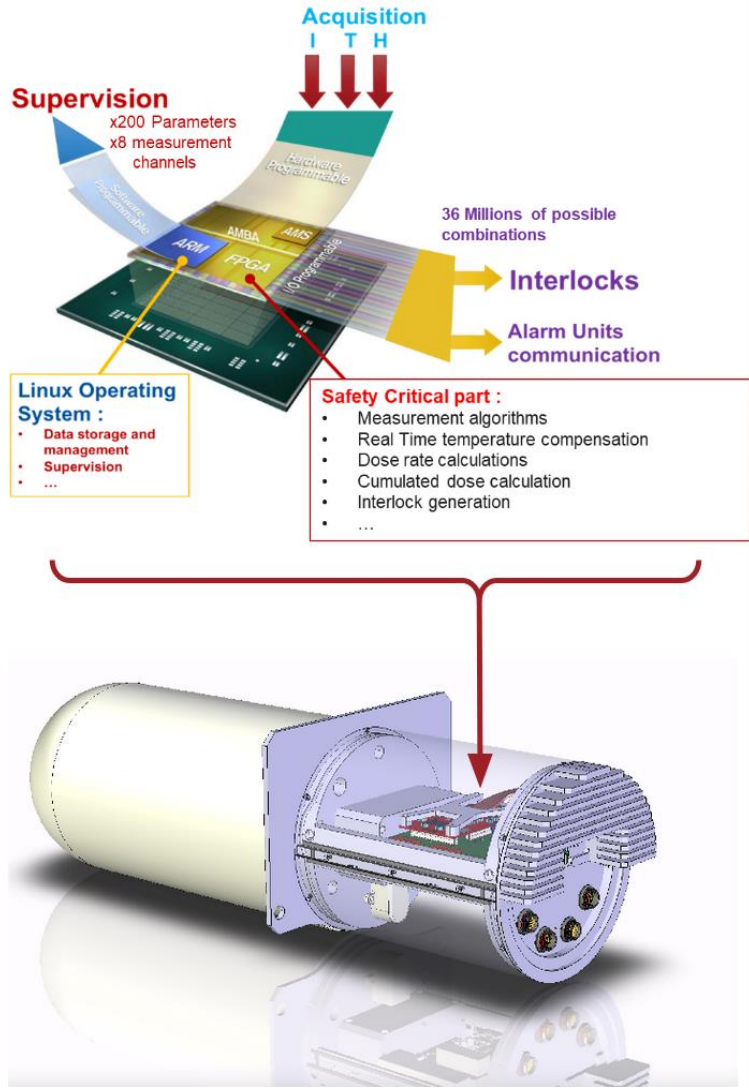
Radiation Monitoring and processing units



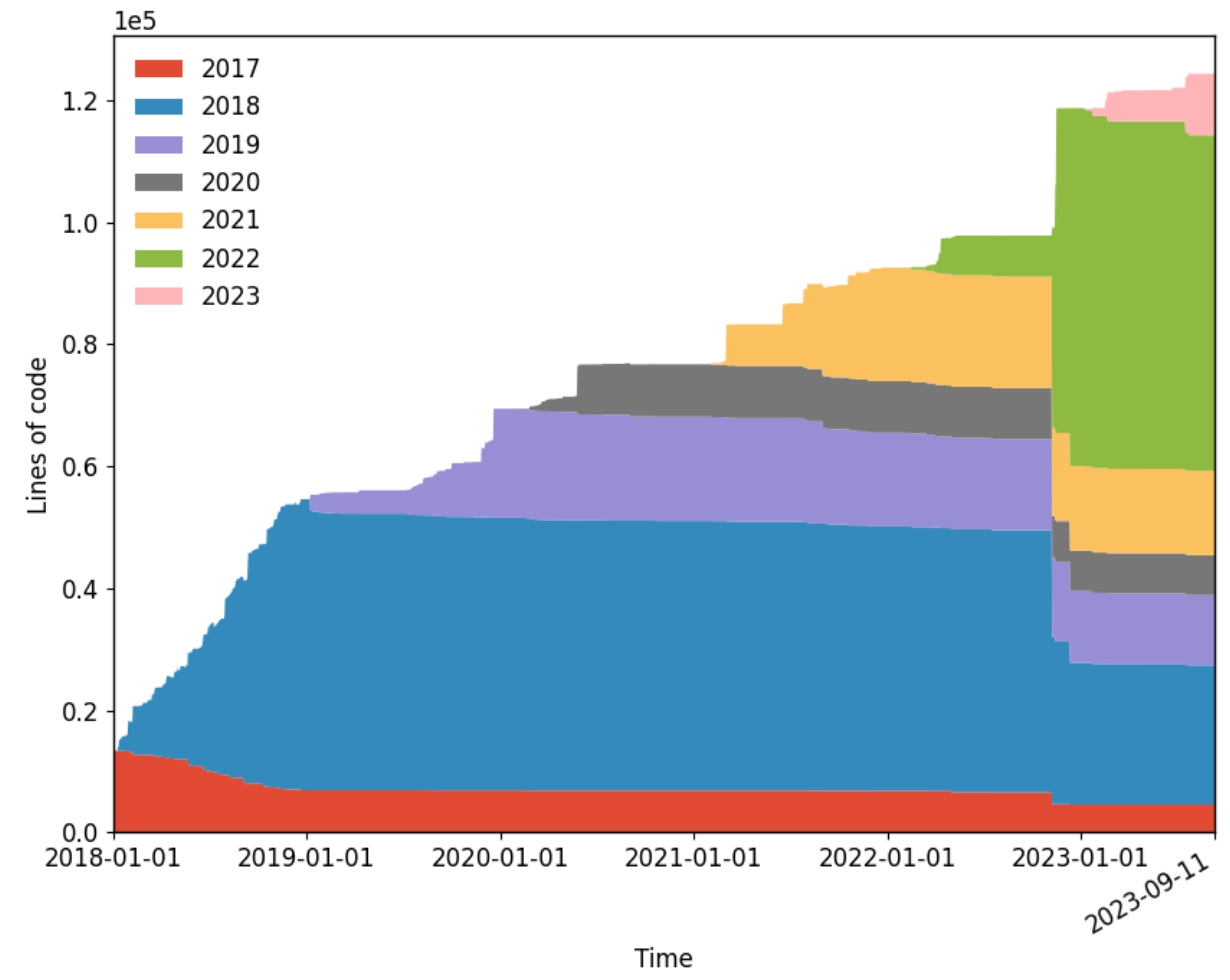
Uninterruptible Power Supply
Includes a battery for continuous operation



CROME Project



SoC Number of Lines of Code (Without CROMiX)



Development Goals

Fast

- Number of compilations
- Avoid debug in hardware

Straightforward / Low entry

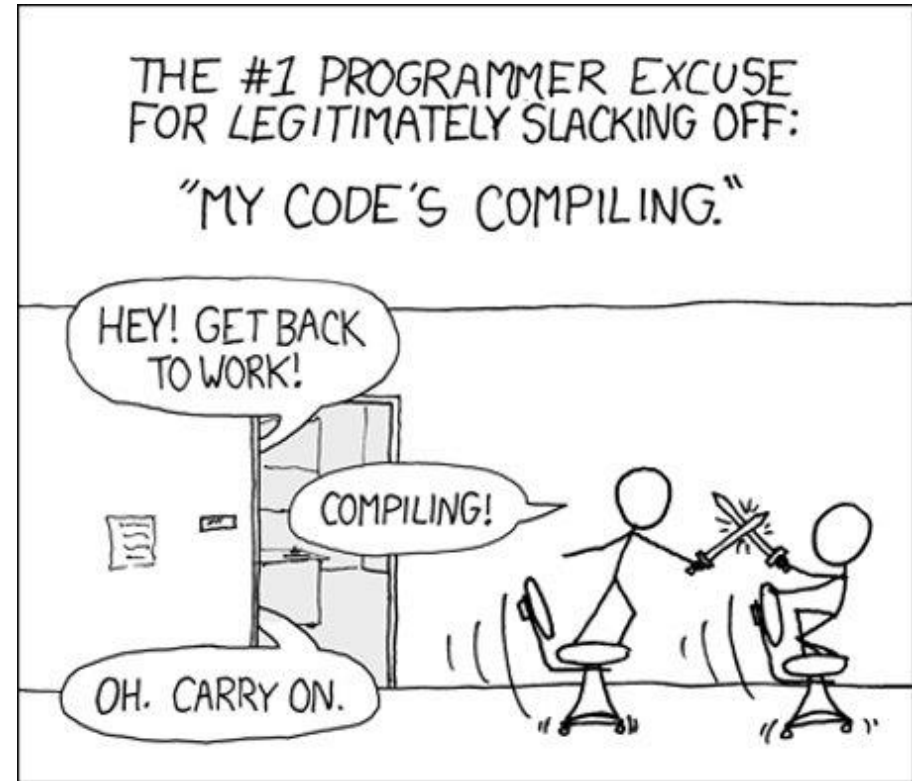
- Accessible, up-to-date documentation
- Simple setup
- Minimal fear driven development

Reliable

- Minimize regressions
- Effective code review

Robust

- Positive contributions must not be bound by the developer's contract time



Main question

How can testing and continuous integration help with our development goals?

Outline:

- Technical Debt
- Testing
- Continuous Integration

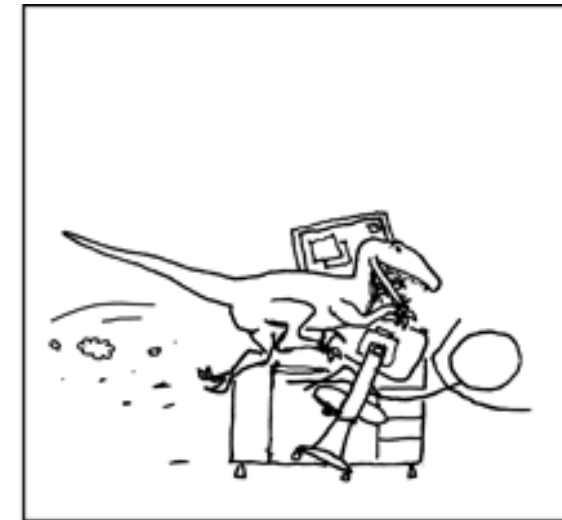
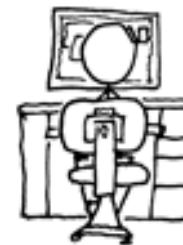
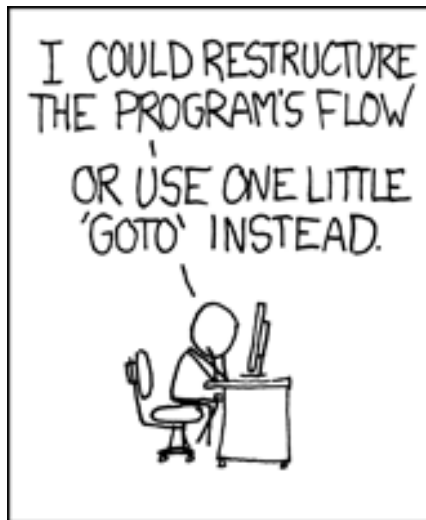


Technical Debt



Technical Debt

In software development, technical debt [...] is the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.



Technical Debt is not star wars

- Neither good nor bad
- Must align with your goals

Re-evaluation:

- New direction
- New functionality
- New tools/technology

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

HOW OFTEN YOU DO THE TASK

	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS

HOW MUCH TIME YOU SHAVE OFF

©xkcd



Testing



If You Didn't Test It, It Doesn't Work

Bob Colwell

- It is much easier than not to write code with bugs
- Any configuration that is not tested is likely to fail in practice



If it's not tested, it doesn't work

- Tests are what allow the design to safely evolve
- Simulation is always cheaper than hardware debugging
- Simulation is more versatile than hardware testing



If it's not tested, it doesn't work

- Test driven development

In our project:

- 1) All (new) code must be tested
- 2) All tests must be added to the repository



Types of tests

- Compilation tests
- Directed tests
 - Behaviour match pre-recorded pattern
- Model tests
 - Behaviour match model
- Ground truth tests
 - Results match external metric (not always applicable)
- Formal tests
 - Mathematically prove certain properties of the design



Types of tests

- Compilation tests
- Directed tests
 - Behaviour match pre-recorded pattern
- Model tests
 - Behaviour match model
- Ground truth tests
 - Results match external metric (not always applicable)
- Formal tests
 - Mathematically prove certain properties of the design

Pure VHDL
Cocotb+GHDL

Questa Property
Checking



Testing guidelines

- Test results "must" be digital
 - Helps review process
- Tests should not peek inside the module
 - Helps future development
- Tests should be quick
 - Quick feedback while developing



Development Goals

Fast

- Number of compilations
- Avoid debug in hardware

Straightforward / Low entry

- Accessible, up-to-date documentation
- Simple setup
- Minimal fear driven development

Reliable

- Minimize regressions
- Effective code review



Robust

- Positive contributions must not be bound by the developer's contract time




Development Goals


Fast

-  Number of compilations
-  Avoid debug in hardware

Straightforward / Low entry

- Accessible, up-to-date documentation
- Simple setup
-  Minimize fear driven development

Reliable

-  Minimize regressions
-  Effective code review

Robust

- Positive contributions must not be bound by the developer's contract time



Continuous Integration

through gitlab CI

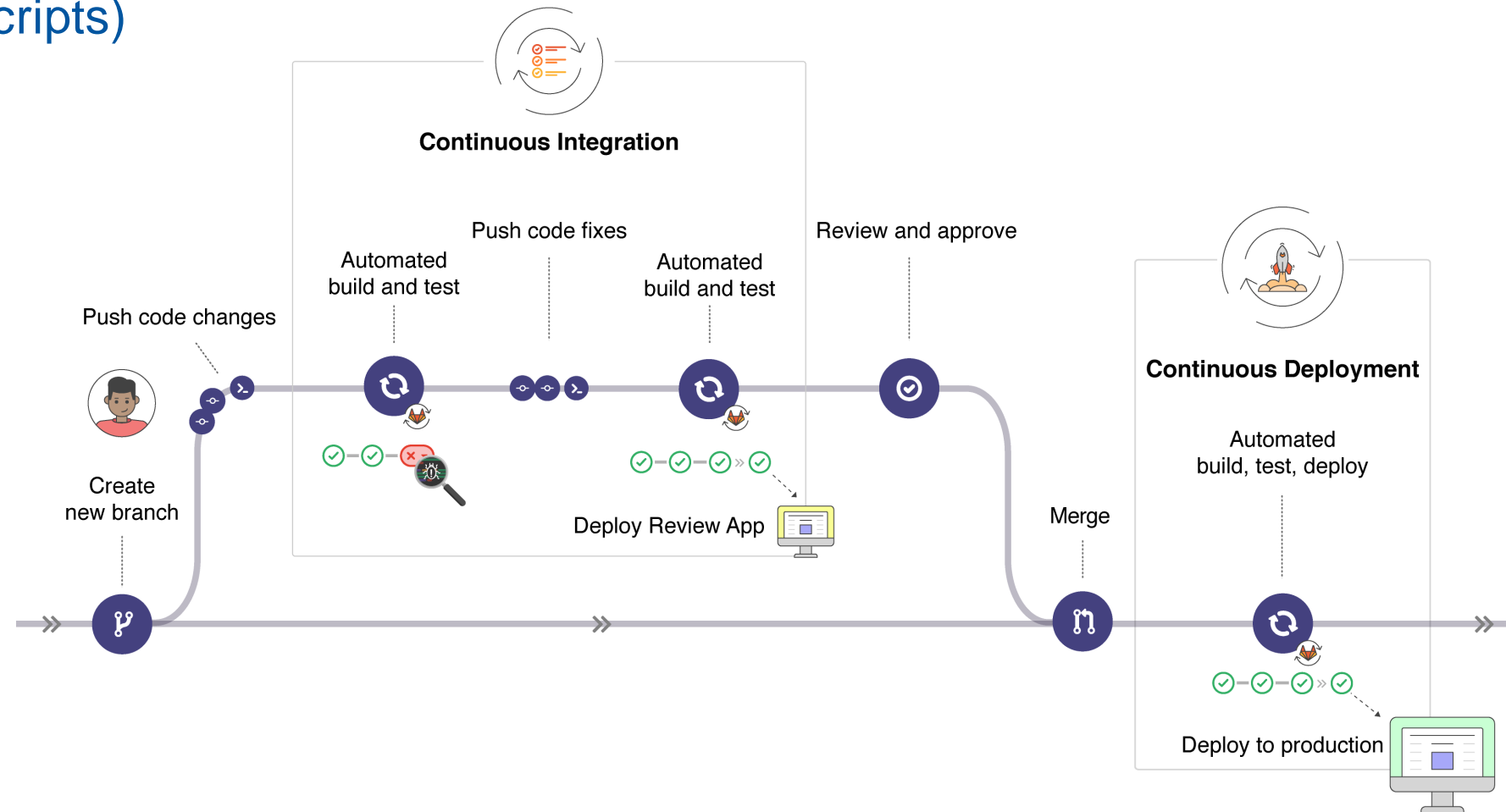
More hands on approach at
<https://indico.cern.ch/event/1208190/>



Gitlab CI

For each proposed code change, automatically run a pipeline (= set of scripts)

- Testing
- Build
- Deployment
- Etc...



Gitlab CI

Advantages:

- Centralization
- Unsupervised build/tests
- Lowers barrier to entry
- Saves results (artifacts) for later inspection
- Simplifies code review



Gitlab CI – Minimal Example

stages:

- build

simple build:

stage: build

image: vivado:2018.1

script:

- make

artifacts:

paths:

- output.hdf



Gitlab CI – Uses in CROME Project

What we use gitlab CI for:

- VHDL cocotb tests
- VHDL style check
- Vivado build
- Documentation build
- SW build
- Petalinux build

What we would like to add:

- SW test suite
- SW style check
- VHDL formal tests
- VHDL UVM tests
- Target multiple hardware revisions
- On-device tests
- ...



Gitlab CI

Friday, September 8

- A** CERN GitLab 18:37 *CROME | Fixed pipeline for accurate/lambda1-update | 0d7a2de0*
- A** CERN GitLab 17:45 *CROME | Failed pipeline for accurate/lambda1-update | 4016f40d*
- A** CERN GitLab 15:15 *CROME | Failed pipeline for accurate/lambda1-update | 14c572a5*

Pipeline Needs Jobs 15 Tests 94 Code Quality

Summary

94 tests 0 failures 0 errors 100% success rate 32.10s

Jobs

Job	Duration	Failed	Errors	Skipped	Passed	Total
check cocotb compile	15.58s	0	0	0	13	13
check_linting	0.00ms	0	0	0	67	67
check cocotb quick	16.52s	0	0	0	14	14

check	build	buildReports	buildImage
check cocotb compile	build ROMULUSlib doc	build hw report usage	build CROMiX
check cocotb quick	build cromeSuite doc		
check cocotb slow	build hw doc		
check_linting	build_hw		
check_vhdl_syntax	build_sw		





Gitlab CI – What I wish was easier

- Making containers for all the tools
 - Shout-out to Adrian Byszuk (SY-EPC-CCE)
- Setting up runners/VM configuration
 - IT support soon?
- Integrating with current HDL development tools
 - More machine readable formats
- Licenses
 - Shout-out to GHDL and cocotb!



Development Goals

Fast

-  100 Number of compilations
-  Avoid debug in hardware

Straightforward / Low entry

- 100 Accessible, up-to-date documentation
- 100 Simple setup
-  Minimize fear driven development

Reliable

-  Minimize regressions
-  100 Effective code review

Robust

- 100 Positive contributions must not be bound by the developer's contract time





Conclusion



Development Goals

Fast

-  100 Number of compilations
-  Avoid debug in hardware

Straightforward / Low entry

- 100 Accessible, up-to-date documentation
- 100 Simple setup
-  Minimize fear driven development

Reliable

-  Minimize regressions
-  100 Effective code review

Robust

- 100 Positive contributions must not be bound by the developer's contract time



Conclusion

- Strict testing guidelines are a requirement
- Investing time in gitlab CI paid off quickly
- Gitlab CI is an invaluable tool for our project



Thank you!



Questions?

