DEPENDABLE DESIGN FLOW FOR PROTECTION SYSTEMS USING PROGRAMMABLE LOGIC DEVICES

M. Kwiatkowski*, B. Todd†, CERN, Geneva, Switzerland

Abstract

Programmable Logic Devices (PLD) such as Field Programmable Gate Arrays (FPGA) are becoming more prevalent in protection and safety-related electronic systems. When employing such programmable logic devices, extra care and attention needs to be taken. The final synthesis result, used to generate the bit-stream to program the device, must be shown to meet the design's requirements. This paper describes how to maximize confidence using techniques such as Formal Methods, exhaustive Hardware Description Language (HDL) code simulation and hardware testing. An example is given for one of the critical functions of the Safe Machine Parameters (SMP) system, used in the protection of the Large Hadron Collider (LHC) at CERN.

CERN is also working towards an adaptation of the IEC-61508 lifecycle designed for Machine Protection Systems (MPS), and the High Energy Physics environment, implementation of a protection function in FPGA code is only one small step of this lifecycle.

The ultimate aim of this project is to create generic techniques and methods applicable to any PLD based system requiring a rigorous implementation and verification.

CERN AND THE LHC

The Large Hadron Collider is the world's most powerful particle accelerator. To reach the new frontiers of physics a centre of mass collision energy of 14 TeV is needed, giving a stored beam energy over 100 times higher than in any other machine (Figure 1).

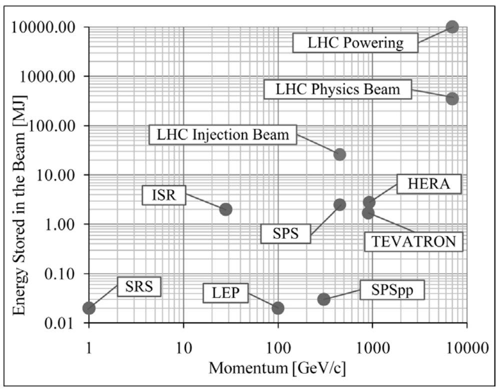

Figure 1: LHC Stored Energy versus other HEP machines [1].

* maciej.kwiatkowski@cern.ch
† benjamin.todd@cern.ch

The LHC is designed to accelerate two counter-rotating beams of $3.2 \cdot 10^{14}$ protons from an injection energy of 450 GeV to a collision energy of 7 TeV. At design values, the peak energy stored in each beam is equivalent to 362MJ, enough to heat and melt around 500kg of copper. A field of 8.3 Tesla is needed to hold the LHC beam in 27km circumference of the machine, this magnetic field is generated by 1232 super-conducting dipole magnets, each having a forward current of almost 13kA, being maintained less than two degrees above absolute zero (-273 degrees Celsius) in a bath of superfluid helium. At design energy, the total stored energy in the LHC magnet powering system is around 10GJ, and a loss of only $10^{-8} - 10^{-7}$ of the nominal beam [2] into one of the superconducting magnets will lead to a quench, where the magnet heats up, becomes resistive and must be switched off to prevent damage.

A complex MPS has been designed to mitigate the risks due to stored beam and magnet energy, a fundamental part of the MPS is the Safe Machine Parameters Controller (SMPC).

SAFE MACHINE PARAMETERS

For the correct protection of the LHC and its accelerator complex, several parts of the MPS require information about the machine's operational parameters. Values such as beam intensities, machine energies, squeezing factors, amongst several others, must be broadcast around the accelerator complex to correctly configure the MPS, and sent to the extraction interlock systems to ensure the correct interlocking of beam transfer between the Super-Proton Synchrotron (SPS) and the LHC.

These parameters are referred to as Safe Machine Parameters (SMP), as they must be generated and distributed

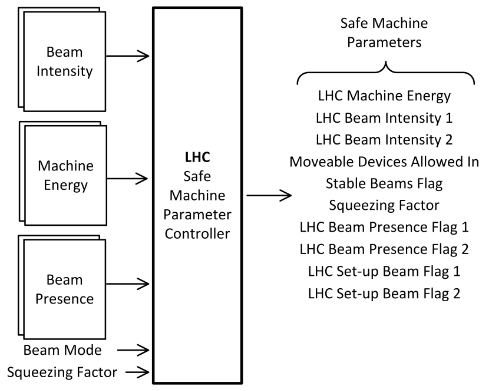

Figure 2: Safe Machine Parameters System.

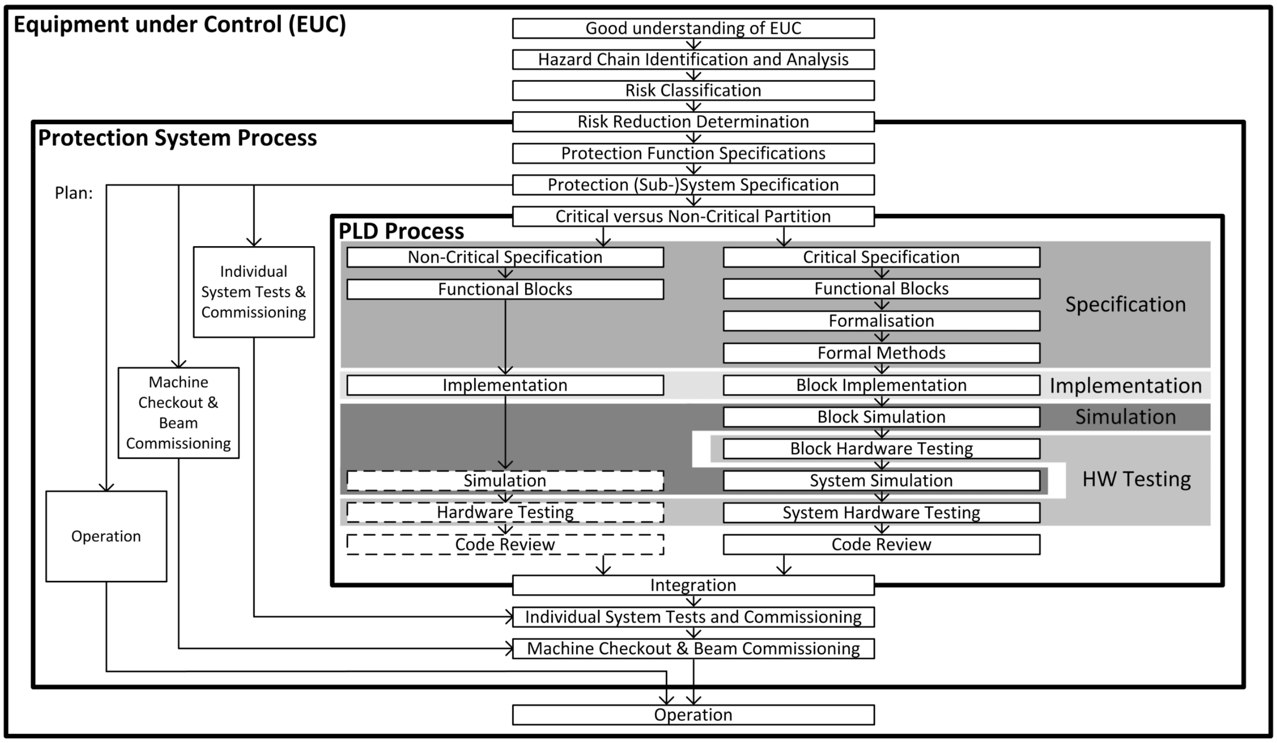
Equipment under Control (EUC)
Good understanding of EUC
Hazard Chain Identification and Analysis
Risk Classification
Risk Reduction Determination
Protection System Process
Protection Function Specifications
Protection (Sub-)System Specification
Plan:
Critical versus Non-Critical Partition
PLD Process
Individual System Tests & Commissioning
Machine Checkout & Beam Commissioning
Operation
Non-Critical Specification
Functional Blocks
Implementation
Simulation
Hardware Testing
Code Review
Critical Specification
Functional Blocks
Formalisation
Formal Methods
Block Implementation
Block Simulation
Block Hardware Testing
System Simulation
System Hardware Testing
Code Review
Specification
Implementation
Simulation
HW Testing
Integration
Individual System Tests and Commissioning
Machine Checkout & Beam Commissioning
Operation

Figure 3: Machine Protection System Lifecycle Based on IEC-61508 E/E/PE Lifecyle.

around the accelerator complex with high dependability (safety, availability and reliability). The SMPC was developed to derive these SMP, taking information from several source systems and providing it to client system, as shown in the Figure 2.

The outputs of the SMPC either go directly to the extraction interlock controllers, or are broadcast around the machines using the General Machine Timing (GMT) network. These different parameters address two principle types of risk: risks during the extraction and transfer of beam from the the SPS into the LHC, and risks during LHC phases following the injection process.

The first group presents particularly tight requirements in terms of time and accuracy, as the transfer of beam from the SPS to the LHC is a very fast process, a single extraction of beam from the SPS is already capable of quenching and damaging LHC magnets.

The SMPC plays a key role in the protection of the LHC and its injector complex.

MACHINE PROTECTION SYSTEM LIFECYCLE

Requirements for the MPS, including the SMPC, came from decades of work and substantial investigations into the performance of the LHC and its MPS, following a deep-thinking argumentative approach. The risks due to LHC stored beam and magnet energy do not pose a threat to personnel or the environment, as as such are not a hard, legal requirement of the LHC project. Nevertheless, there are significant advantages to be gained by considering the development of the MPS as if it were a safety system, being legally required.

During the development of the SMPC, and some of the other MPS sub-systems, such as the Beam Interlock System and Powering Interlock Controllers, common themes and ideas have began to emerge. The combination of these ideas has led to the concept of a MPS Lifecycle (MPSL), based on the IEC 61508 overall system lifecycle. The MPSL concept allows a consistent approach to the evaluation of existing parts of the MPS, whilst at the same time providing a framework for the development of new protection systems. Figure 3 shows this proposed MPSL.

The part of the MPSL which is relevant to the programmable logic device flow is that related to the implementation of so-called Protection Functions (PFs) which are being established using devices such as FPGAs. The following sub-sections detail some of the key concepts of the MPSL, before concentrating on those parts relevant for PLDs.

Equipment Under Control

The Equipment Under Control (EUC) must be well understood by protection system designers, in CERN's case, the EUC can be considered as the particle accelerator and its associated mechanical, electric and electronic equipment. To realise such complex systems, experts from many domains must work together to establish potential hazards and the effects these have on the EUC. Based on this risk analysis approach, each risk should be assessed and classified. Risk being a combination of the probability of occurrence and the severity of a hazardous event and it is usually expressed qualitatively (qualitative risk analysis).

Protection System Process

The designers are then to conceive mitigations for these risks by working on two fronts: working to reduce the likelihood of the hazardous event, and by working to reduce the consequences of such an event were it to occur. The ultimate goal of this is to reduce risks to acceptable levels, another way is to use the phrase "As Low as Reasonably Possible" (the ALARP principle). This means that for each risk that is identified, a reduction is needed to achieve the ALARP level. The higher the risk the larger the required Risk Reduction Level (RRL).

Several approaches can be used to reduce risks, some of which result in the creation of PFs. It is possible that several risks can be mitigated by many PFs, and the PFs can be spread across many sub-systems.

In the case of the LHC MPS, one of these sub-systems is the SMP system, which is required to implement several PFs, by exploiting PLDs.

PROGRAMMABLE LOGIC DEVICE PROCESS

The PLD design process starts with clear separation of what is part of the PFs from other non-critical functions. Functions that are part of the PF are to follow this PSL process, whereas those functions that are non-critical are not needed to have the same levels of rigour. The conceptual separation of the critical from non-critical functions should be followed through to the hardware realisation. Hardware must not mix critical and non-critical functions! If no other choice exists, then non-critical functions which share common devices with critical ones must also be verified using the full protection system process.

In case of the SMPC each printed circuit boards has two separate FPGA devices for functions. The first, a control FPGA device, is tasked with critical functions, the second, a monitor device, records and supervises the operation of the control device and implements the non-critical functions. For example, the online monitoring of the SMPC, which is important but not critical for the protection of the EUC.

Specification Phase

Each critical function must be decomposed into functional blocks, ones which lend themselves easily to analysis and understanding. In addition, each must be capable of being readily specified using a formal language.

For example, the block diagram of the SPS Setup Beam Flag (SBF) is shown on Figure 4. This flag is used in the process of beam injection from the SPS to the LHC accelerator. The SPS SBF is a function of SPS beam intensity, to increase availability the intensity information is redundant,

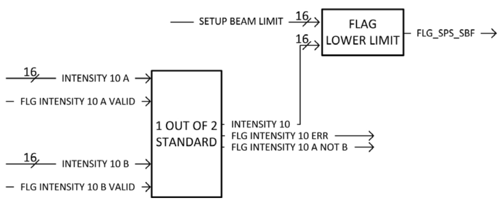

Figure 4: Setup Beam Flag blocks example.

being merged into one value. The one out of two block selects valid intensity value, when both of the sources are not valid the fail safe value is applied. The limit block compares calculated intensity value to a predefined threshold (SETUP_BEAM_LIMIT).

The formal specification of the SPS SBF calculation is presented on Figure 5. Error and selection flags (FLG_INTENSITY_10_ERR, FLG_INTENSITY_10_A_NOT_B) are not formalized because they are used for the monitoring purpose and they do not belong to the critical functionality. The key strength of this Predicate Logic formal language is in interpretation: if correctly written, it can only be understood by designers in a single way, which is not always the case for a traditional specification. The formal language also allows a formal verification of some design parameters, allowing a mathematical verification the completeness and the consistency of the specification.

```
FAIL_SAFE_HIGH_INTENSITY : int := 6.5535 * 10^14;

SPS_SMPC_SBF (
  SETUP_BEAM_LIMIT : int,
  INTENSITY_10_A : int,
  FLG_INTENSITY_10_A_valid : bool,
  INTENSITY_10_B : int,
  FLG_INTENSITY_10_B_valid : bool,
  FLG_SPS_SBF : bool
  ) :=
    exists INTENSITY_10 : int. (
    (INTENSITY_10 =
      if FLG_INTENSITY_10_A_valid then INTENSITY_10_A;
      if else FLG_INTENSITY_10_B_valid then INTENSITY_10_B;
      else FAIL_SAFE_HIGH_INTENSITY;
    )
    AND
    FLG_SPS_SBF =
      if INTENSITY_10 <= SETUP_BEAM_LIMIT then TRUE;
      else FALSE
  );
```

Figure 5: Setup Beam Flag in Predicate Logic.

Implementation Phase

Each of the smaller functional blocks which were conceived in the previous step are then implemented. Each block should be the correct size as to be simulated completely before being integrated into the sub-system. When all the blocks are implemented and have passed simulation, the complete system can be composed by connecting the blocks together. The actual implementation phase is a very small part of the overall time spent on PLD firmware development. The vast majority of the time is spent on simulation and testing. Hardware Description Language

(HDL) code, such as VHDL or Verilog, is written to describe PLD function, HDL languages should not be associated with programming languages but rather with modeling tools. The designer should always know what the expected result of the synthesis process is to be, this should be also verified using the output of synthesis tools. This kind of verification is much easier when the blocks are small and readily understood.

Software Simulation Phase

Software simulation is carried out on the block level as well as on the system level. Simulation with code coverage is a fundamental requirement for critical functions. A software test-bench is required, which should wrap the Unit Under Test (UUT) inside Bus Functional Models (BFMs), passing stimuli to the UUT and recording its responses. Behaviour which is not specified for the block or the subsystem can be detected and fixed, at the same time the testbench should evolve to include new conditions as the weaknesses in code coverage are identified. It is preferred that a critical function achieves full code coverage.

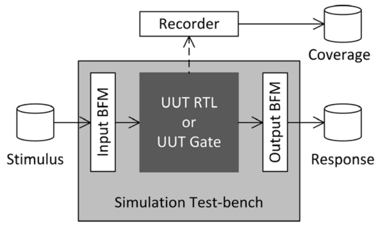

Figure 6: Software simulation with code coverage.

Hardware Testing Phase

Once the smaller blocks have been implemented, simulated and combined, the real hardware can be generated. This is then tested using a dedicated hardware tester. Hardware testing is obligatory on the system level but optional at the block level, this is due to technical difficulties in realising good test equipment and also the cost of the hardware tester construction must be considered. On the other hand, a complex block may warrant the investment of time and money in a dedicated test hardware.

The hardware tester generates input stimulus and checks the response of the Device Under Test (DUT), in much the same way as the simulation test-bench, but this time using real signals, logging real results. Its advantage over the software simulation is its speed and the possibility to introduce real distortions, such as noise on a signal. A very useful hardware testing tool provided by device manufacturers is an Embedded Logic Probe (such as Xilinxs ChipScope or Alteras SignalTap). The probe is integrated with the design and finally programmed into PLD DUT. It uses device memory resources for recording selected internal or external signals, thus the critical device must have spare cells and memory to accommodate this. With this analyzer in place it is possible to record internal signals using a variety of trigger conditions. A typical setup with an embedded logic probe is shown on Figure 7.

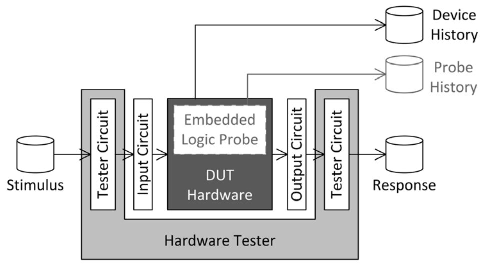

Figure 7: Hardware tester.

Code Review

Finally, once all elements have been proven to function as expected, the HDL code should be reviewed via at least an internal audit, and preferable including an external audit too. A key element during this phase is good documentation and especially a clear depiction of the formalised version.

CONCLUSIONS

All the phases described above should be documented and kept to form a basic safety case. When bugs or mistakes are found it will help to identify weak points of the test cases, testing processes, and even weaknesses in the design specification. All of this information can be used to correct future implementations. In addition, a clever selection of the design partitioning, and basic functional block requirements vastly increases HDL code reusage: thus the invested time in test benches and verification is also saved, and it is possible to reuse the same well verified blocks in the future designs.

The MPSL presented gathers many elements which have been often used in the development of systems at CERN, but are not necessary currently being done in a complementary and systematic way. These concepts and ideas are very much a work in progress, at the same time it is evident that such an approach encourages both designers, and those that specify designs to use good practices, and really understand what they require of a system. In doing this the confidence in the final system can be increased.

REFERENCES

[1] R. Assmann et al, "Requirements for the LHC collimation system", EPAC'02, La Vilette, Paris, 2002, `http://jacow.org/e02/TALKS/TUAGB001.pdf`.

[2] R. Schmidt and J. Wenninger, "Protection against Accidental Beam Losses at the LHC", PAC'05, Knoxville, 2005, `http://jacow.org/p05/PAPERS/MOPA005.PDF`.