Pedro Afonso DEITOS REGIS

Systèmes Embarqués et Objets Connectés
2022-2023

STMicroeletronics
12 Rue Jules Horowitz, 38019 Grenoble

Lazify - A tool for automated testbench code generation

from 13/02/23 to 11/08/23

Under the supervision of:

- Company supervisor: Thomas, ALOFS, thomas.alofs@st.com

- Phelma Tutor: Katell, ALLORY, katell.morin-allory@univ-grenoble-alpes.fr

Confidentiality: no

Ecole nationale supérieure de physique, électronique, matériaux

Phelma
Bât. Grenoble INP - Minatec
3 Parvis Louis Néel - CS 50257
F-38016 Grenoble Cedex 01

Tél +33 (0)4 56 52 91 00
Fax +33 (0)4 56 52 91 03

http://phelma.grenoble-inp.fr

.

DÉCLARATION DE NON-CONFIDENTIALITÉ

Grenoble, le 23/02/2024

~~<Nom de la ville>, <jour> de <mois> de <année>~~.

En tant que représentant de ST Microelectronics <~~l'institution où le CBP a été réal~~isé> dans laquelle ce travail a été effectué, je déclare que ce document ne contient aucune information confidentielle, sensible ou délicate qui empêcherait sa publication par l'Université fédérale de Santa Catarina (UFSC) en vue d'un accès au grand public, y compris la mise à disposition de ce document en ligne dans le dépôt institutionnel de la bibliothèque universitaire de l'UFSC. En outre, je déclare être conscient que l'auteur, en tant qu'étudiant (ou diplômé) de l'UFSC, est tenu de déposer ce document, étant donné qu'il s'agit d'un travail de conclusion de cours, dans le dépôt institutionnel susmentionné, conformément à la résolution normative n ° 126/2019/Cun.

Comme j'accepte ces conditions, je signe ci-dessous.

T. Alofs

<Untel>

<l'institution où le CBP est détenu>

thomas.alofs@st.com +33.6.37.17.75.56

Verificaton engineer at STMicroelectronics & internship tutor for Pedro Afonso Deitos Regis

from 02/23 → 08/23

Contents

1 Abstract **3**

2 Resume **3**

3 Introduction **4**

3.1 General Context 4

3.2 Restricted Context 4

3.3 Problematic 5

3.4 Objective 6

3.5 Planning 6

4 About the project **7**

5 The Lazify Tool **18**

5.1 First Step - Format Specification 19

5.2 Second Step - Parser 20

5.2.1 Data Base 21

5.2.2 Parser 22

5.3 Third Step - Generating Code. 22

6 Implementing the tool in the STUSB4531 project **23**

7 Conclusion **27**

7.1 Current Limitations and Future Prospects 28

7.2 Impact 29

7.3 Personal Conclusion 29

8 Annexes **30**

8.1 Main Loop Code 30

8.2 Assertions 32

8.3 Reduced Assertions 39

List of Figures

2 Cost of a bug in the conception of product. 6
3 High Level Architecture of the Project 7
4 Main loop 9
5 PPS2PPS loop 11
6 FIX2FIX loop 12
7 VAR2FIX loop 13
8 VAR2AVS loop 14
9 FIX2AVS loop 15
10 Format for the Lazify Tool 19
11 Data Base used in Lazify. 21
12 Main Loop Excel Sheet 25
13 Assertions Excel Sheet 26

List of Tables

1 FIXED RDO 17
2 PPS RDO 17
3 AVS RDO 18

1 Abstract

Lazify is a set of tools designed to streamline and automate the code generation process for reference models that are commonly part of verification testbench for digital systems. It aims to improve the efficiency, versatility and ease of verification tasks. By automating certain aspects of the code generation process, Lazify simplifies the entire verification workflow, making it faster and more generic.

The specific functionalities and features of Lazify may vary depending on the context or application it is being used for. However, in general, Lazify tools provide a systematic approach to extract relevant information from Excel files, parse the data, and generate code, typically in languages such as SystemVerilog.

By leveraging Lazify, Digital Verification Engineers can save time and effort by automating repetitive tasks and reducing the potential for human error. This enables engineers to focus more on higher-level verification activities and analysis, ultimately enhancing the overall verification process.

2 Resume

Lazify est un ensemble d'outils conçus pour rationaliser et automatiser le processus de génération de code pour les modèles de référence qui font communement partie des test-bench de verification pour des systemes digitaux. Il vise à améliorer l'efficacité, la polyvalence et la facilité des tâches de vérification. En automatisant certains aspects du processus de génération de code, Lazify simplifie l'ensemble du flux de travail de vérification, le rendant plus rapide et plus générique.

Les fonctionnalités spécifiques de Lazify peuvent varier en fonction du contexte ou de l'application pour laquelle il est utilisé. Cependant, en général, les outils Lazify fournissent une approche systématique pour extraire les informations pertinentes des fichiers Excel, analyser les données et générer du code, généralement dans des langages tels que SystemVerilog.

En utilisant Lazify, les ingénieurs en vérification numérique peuvent gagner du temps et de l'énergie en automatisant les tâches répétitives et en réduisant le risque d'erreur humaine. Les ingénieurs peuvent ainsi se concentrer davantage sur les activités de vérification et d'analyse de haut niveau, ce qui permet d'améliorer le processus de vérification dans son ensemble.

3 Introduction

3.1 General Context

STMicroelectronics [1] is a global leader in the semiconductor industry, providing innovative solutions for a wide range of applications in various sectors such as automotive, industrial, personal electronics, and communications. Founded in 1987, the company has grown to become one of the largest semiconductor manufacturers in the world, with operations in over 35 countries and more than 45,000 employees. STMicroelectronics is committed to delivering cutting-edge technology and products that enable its customers to stay ahead of the curve in today's fast-paced, ever-changing technological landscape. With a focus on sustainability and corporate responsibility, STMicroelectronics is dedicated to making a positive impact on society and the environment.

STMicroelectronics exceptional revenue growth in 2022, with a 26% increase from the previous year and a net income of $4 billion, a staggering 98% more than the previous year, is a testament to the company's unwavering commitment to quality and innovation. This impressive performance has allowed STMicroelectronics to gain significant market share over its competitors, solidifying its position as a leader in the semiconductor industry. With its strong financial position and continued investment in research and development, STMicroelectronics is well-positioned to continue its growth trajectory in the coming years. As the demand for advanced semiconductor solutions continues to rise, STMicroelectronics is poised to play a leading role in shaping the future of the industry.

The internship took place at STMicroelectronics Grenoble, which is one of the company's largest research and development centers. Located in the heart of the French Alps, the Grenoble site is home to over 3,000 employees and is responsible for the development of cutting-edge technologies and products in various fields such as automotive, industrial, and personal electronics.

3.2 Restricted Context

The internship took place in the AMS division of STMicroelectronicsin a team responsible for verifying various cutting-edge products that are expected to be launched in the near future. Our team utilized advanced technologies and collaborated closely with other departments to ensure that the products met the highest quality standards.

Verification is a crucial process in microelectronics that ensures that a design meets the intended functionality and performance requirements. It involves checking the design against the functional specifications to ensure that it meets the desired functionality and performance requirements before sending the design to the wafer fab for production of the chip. Verification is a complex process that involves multiple steps, including simulation, formal verification, and testing.

Simulation involves running the design through a software simulator based on a RTL model of the circuit. This allows designers to test the functionality of the design and identify any errors or issues that need to be addressed. This is a rigorous process that is typically used for safety-critical applications.

Testing involves physically testing the design to ensure that it meets the intended functionality and performance requirements. This involves creating test cases that stress the design and identify any issues that need to be addressed. Testing is typically done using specialized equipment that can measure the performance of the circuit.

Overall, verification is a critical process in microelectronics that ensures that designs meet the intended functionality and performance requirements. It is a complex process that involves multiple steps, including simulation, formal verification, and testing. By following a rigorous verification process, designers can ensure that their designs are reliable, safe, and perform as intended.

3.3 Problematic

Writing code for functional verification is necessary because it allows engineers to simulate the behavior of the product under various conditions and configurations. This is particularly important for complex products that have multiple functions and interactions. By writing code, engineers can test the product's behavior in a controlled environment and identify any functional issues that may have been introduced by the RTL designers.

The code used in the verification process is typically written in a hardware description language (HDL) such as Verilog or VHDL. These languages are used to describe the behavior of the electronic components and systems being designed. However, writing code for verification usually is a manual and time-consuming and challenging task. It requires a deep understanding of the product's design and functionality, as well as knowledge of programming languages and testing methodologies. Moreover, the verification code must be constantly updated and refined as the product evolves, which adds an additional layer of complexity to the verification process.

Rewriting the code used in the verification process can be problematic for several reasons. The verification process is critical to ensuring the reliability and safety of the electronic systems being designed. Any errors or mistakes in the verification process can lead to serious consequences, such as system failures or safety hazards.

Furthermore, rewriting the code may require significant time and effort. This can result in additional costs and delays in the design process.

Despite the challenges, writing code for verification is essential for ensuring the quality and reliability of a product. It allows engineers to identify and fix issues early in the development process, which can save time and resources in the long run. As technology continues to advance and products become more complex, the need for effective verification methods will only continue to grow.

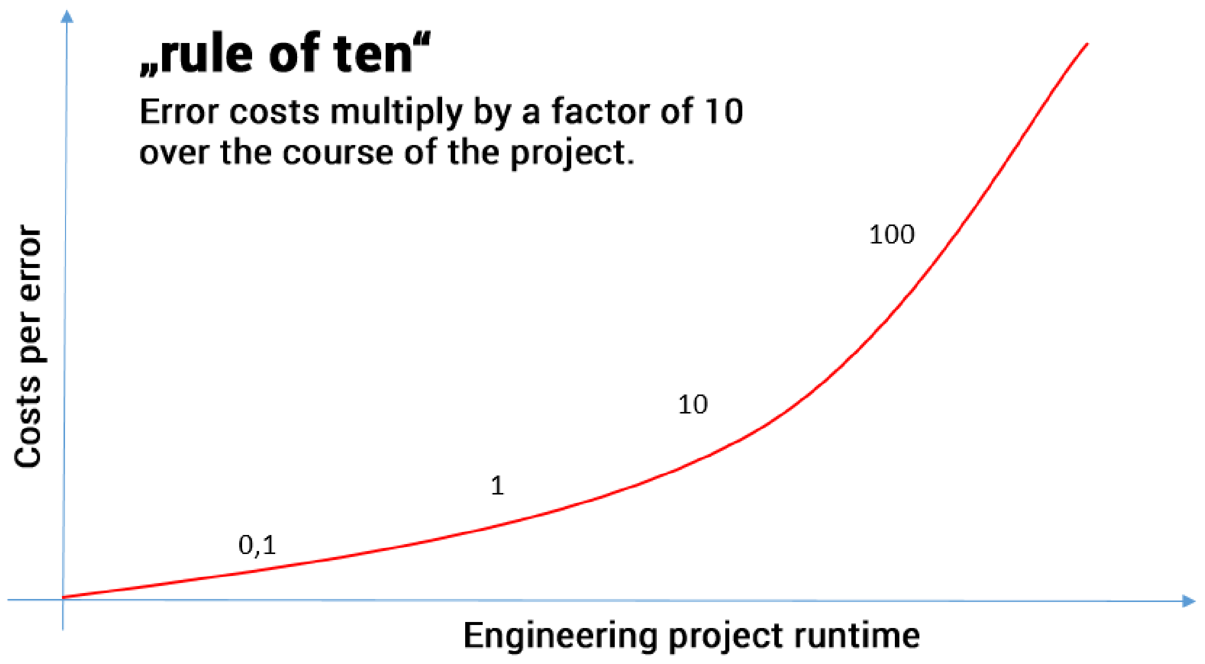


Figure 2: Cost of a bug in the conception of product.

Finding bugs in the code during the verification phase is a critical and essential part of the software development process. Verification, also known as testing or quality assurance, involves systematically evaluating the design to ensure it meets its specified requirements and functions correctly. The importance of finding bugs during this phase cannot be overstated, as it directly impacts the overall quality, reliability, and security of the software.

3.4 Objective

Considering this, the verification team at STMicroelectronics contemplated initiating a project aimed at developing a software solution capable of automating the code generation process, intended to speed-up verification tasks. The Lazify primary objectives encompass maximum versatility, speed, and user-friendliness, minimizing the necessity for manual coding. Moreover, it should be readily deployable and exhibit a gentle learning curve for users.

Furthermore, Lazify has the potential to assist verification engineers by offering a well-defined verification strategy. Through tool utilization, users would input information in an intelligible format, thus facilitating the creation of a comprehensible verification plan.

3.5 Planning

Following discussions with various teams, tutors, and internal team deliberations, we have collectively reached the consensus that the optimal strategy entails dividing the project into three key focal points. This delineation could be aptly termed a "three-step approach" to address the issue at hand.

- Translating the specification from the design team into a common input format specified for Lazify.
- Using a parser to parse the information for the input format into a common database that will later be used.
- Writing custom scripts that will exploit the common database to generate the code.

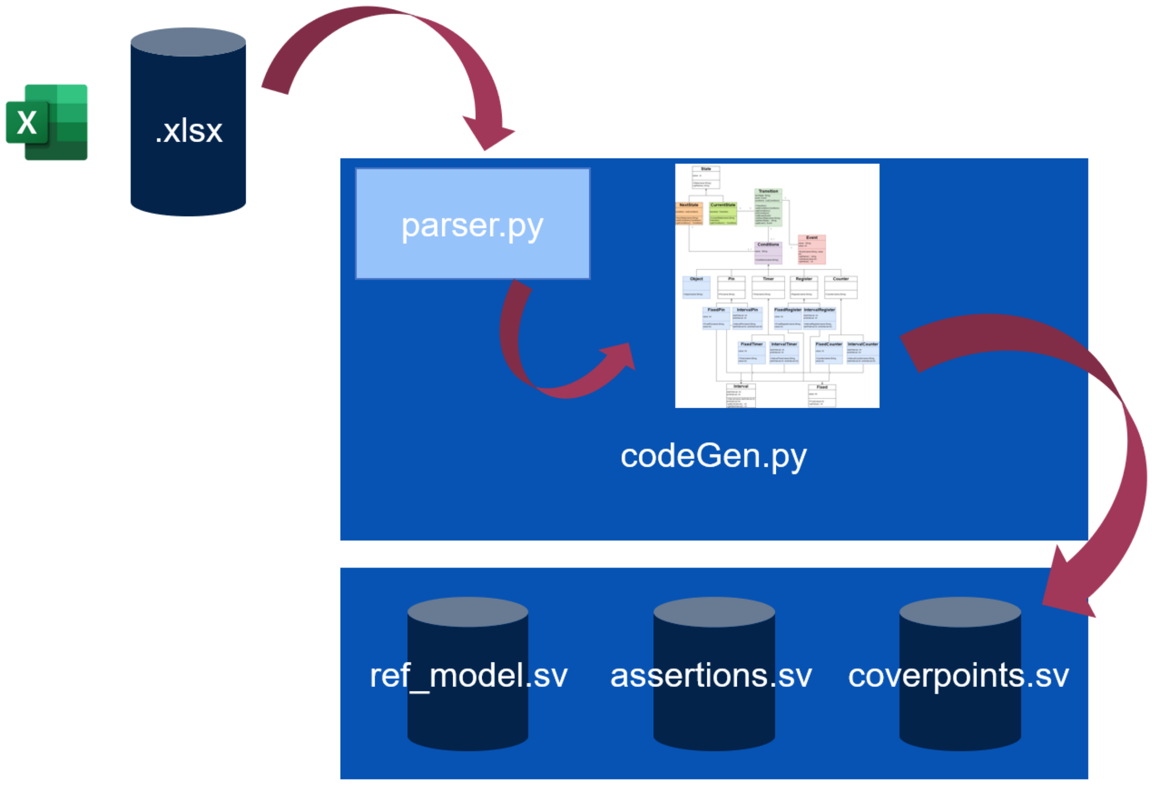


Figure 3: High Level Architecture of the Project

The three-step methodology is illustrated in Figure 3. Initially, the selected format for this project is the Excel (.xlsx) format. Next, within the codeGen python script, there exists a parser responsible for collecting data from the Excel input document. Subsequently, custom scripts come into play, leveraging this generated database to automate code generation.

4 About the project

The endeavor occurred within the framework of the STUSB4531 project, aimed at creating a chip for USB Type-C PD (Power Delivery) application. This innovative product centers around rapid charging and a dynamic PD protocol harnessed through the USB

Type-C connector. By leveraging this technology, devices can receive elevated power levels, resulting in accelerated charging and enhanced power flexibility.

The USB Type-C PD protocol facilitates power negotiation, permitting the provisioning of elevated voltage and current rates for charging. This proves particularly advantageous for power-intensive devices like laptops, tablets, and smartphones.

The main functionally that will be explained is the Power Negotiation and Delivery, the DPM block is involved in the negotiation and management of power delivery between a USB Type-C host and a connected device. This includes determining power requirements, negotiating power contracts, and monitoring power delivery to ensure safe and efficient charging. This negotiations and management of the power delivery happen with the help of PDOs and RDOs.

In the context of USB Power Delivery (USB PD) and the Device Policy Manager (DPM) block within the USB Type-C standard, Power Data Objects (PDOs) and Request Data Objects (RDOs) are fundamental components that facilitate the negotiation and management of power delivery between USB Type-C devices.

Power Data Objects (PDOs):

- Definition: PDOs represent the different power profiles or configurations that a USB Type-C device is capable of supplying or accepting. Each PDO specifies a combination of voltage levels, current limits, and other parameters that define a particular power delivery capability.
- DPM Role: The DPM block is responsible for evaluating and managing the PDOs of a USB Type-C device. It determines the available power options based on the device's capabilities and communicates this information to other connected devices during the negotiation process.
- Negotiation: When two USB Type-C devices are connected, they exchange their respective PDOs. The DPM of each device evaluates the PDOs received from the other device to determine the best-suited power delivery configuration.

Request Data Objects (RDOs):

- Definition: RDOs are messages sent by a receiving USB Type-C device (typically a Sink) to a transmitting device (typically a Source) to request a specific power delivery profile. An RDO specifies the desired voltage, current, and other parameters.
- DPM Role: The DPM block, specifically on the Sink side, generates RDOs based on the device's power requirements and desired operating conditions. It sends these RDOs to the Source device to initiate the negotiation process for power delivery.
- Negotiation: The Source device's DPM receives the RDO and evaluates whether it can fulfill the requested power profile while considering its own PDOs and power capabilities. The negotiation may involve sending Accept or Reject messages back to the Sink until an agreement is reached on a compatible power delivery configuration.

The DPM block dynamically manages the negotiation process by comparing the PDOs and RDOs to find the best common power profile that both devices can support.

The negotiation aims to optimize power delivery efficiency while meeting the power requirements of the receiving device. This allows devices to operate at the highest power level possible without exceeding safety limits. Throughout the negotiation process, the algorithm's objective is to discover a match, which is know as PDO match, wherein the values within the match satisfy the criteria of both the Sink and the Source.

The DPM continuously monitors the power delivery process, making adjustments as needed, such as during changes in load conditions or power source availability.

In summary, the DPM block plays a crucial role in managing the negotiation between USB Type-C devices by evaluating Power Data Objects (PDOs) and generating Request Data Objects (RDOs) to establish an optimal power delivery configuration. This RDO value is being written in the ALGO_RESULT register. This dynamic negotiation ensures efficient and safe power delivery while adapting to the capabilities and requirements of the connected devices.

To produce the correct RDO there are different types of loops that are selected base on the state of the Main loop presented in the figure 4

Please note that certain parts of this text have been intentionally masked or hidden for confidentiality reasons. This is to protect sensitive information from unauthorized access or disclosure. Thank you for your understanding

Figure 4: Main loop

- src_cull_null_ff: the maximum power of the Source (PDP_SRC) is null
- end_loop_no_match: no PDO match found
- src_avs_ignore: ignore AVS APDO (Device setting from the registers block)

If the device receives a $i_p e_n ew_c ontract_r eq$ the algorithm goes to the state $save_s rc_p dos$, in this state the algorithm:

- saves the SRC_PDOj received from the SRC in the registers

- extract the PDP_SRC information

After that the algorithm might go to PPS2PPS or FIX2FIX depending if the PPS has priority of not.

In case it enter in the PPS2PPS loop the algorithm, it:

- enables the secondary FSM to compare the PPS SPR SNK_PDO with the PPS SPR SRC_PDOj (which typically only consists of one AVS SPR)
- waits for the secondary FSM to generate a pdo_match pulse if a match is found and an PPS RDO is prepared, else if there is no match found

In case it enter in the FIX2FIX loop the algorithm, it:

- enables the secondary FSM to compare the Fixed SNK_PDOi (up to three PDOs) with the Fixed SRC_PDOj (up to seven PDOs)
- waits for the secondary FSM to generate a pdo_match pulse if a match is found and a FIX RDO is prepared, else if there is no match found, an end_loop_no_match pulse is generated instead

For the VAR2FIX loop the algorithm, it:

- enables the secondary FSM to compare the Variable SNK_PDOi with the Fixed SRC_PDOj (up to seven PDOs)
- waits for the secondary FSM to generate a pdo_match pulse if a match is found and a FIX RDO is prepared, else if there is no match found, an end_loop_no_match pulse is generated instead

For the FIX2AVS loop the algorithm, it:

- enables the secondary FSM to compare the Fixed SNK_PDOi (up to three PDOs) with the AVS SPR SRC_PDOj (which typically only consists of one AVS SPR).
- waits for the secondary FSM to generate a pdo_match pulse if a match is found and an AVS RDO is prepared, else if there is no match found, an end_loop_no_match pulse is generated instead

For the VAR2AVS loop the algorithm, it:

- enables the secondary FSM to compare the Variable SNK_PDO with the AVS SPR SRC_PDOj (which typically only consists of one AVS SPR).
- waits for the secondary FSM to generate a pdo_match pulse if a match is found and an AVS RDO is prepared, else if there is no match found, an end_loop_no_match pulse is generated instead

And for the REQUEST_RDO state:

- a pulse is generated that indicates to the PE (Policy Engine) that the Best_RDO is ready.

For the secondary FSM that are a few possible states:

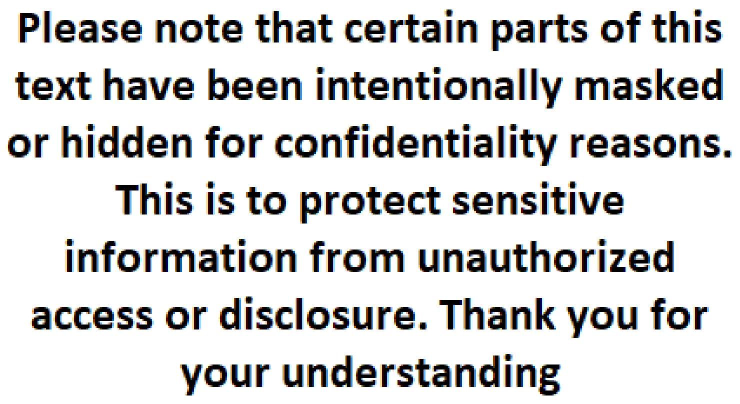
- FIND_V_MATCH state: finds a voltage match between the SNK and the SRC
- CHECK_CURR state: finds a current match between the SNK and the SRC

- PDO_MATCH state: generates the pdo_match and prepares a RDO to the Main FSM
- IDLE_RDO state: idle state reached after the PDO_MATCH state generates the pdo_match

It is important to note that the index of the variables signify the position in the vector of the current or voltage being compared at the moment.

Figure 5: PPS2PPS loop

As we can see in the figure 5 the algorithm assesses whether the chosen voltage falls within the range defined by the source's minimum and maximum voltage limits. Furthermore, it validates whether these proposed source limits align with the boundaries of the sink's minimal and maximal PPS values. When all these conditions are satisfied, the algorithm proceeds to compare the sink's maximum current with that of the source. Based on this comparative analysis, the RDO value undergoes modification accordingly.



Please note that certain parts of this text have been intentionally masked or hidden for confidentiality reasons. This is to protect sensitive information from unauthorized access or disclosure. Thank you for your understanding

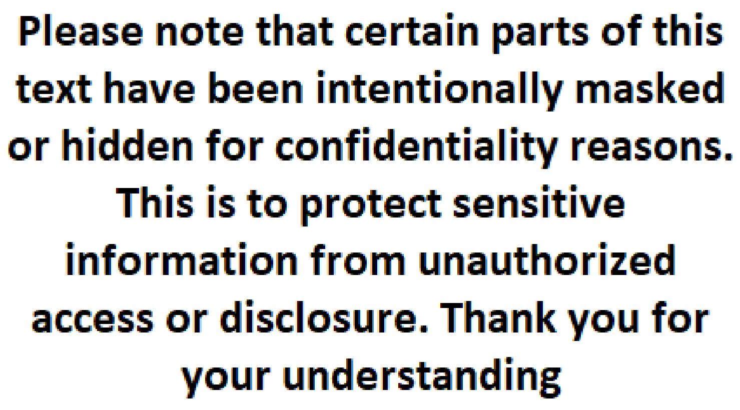
Figure 6: FIX2FIX loop

As we can see in the figure 6 the algorithm evaluates whether the current values within the predetermined fixed PDOs of both the source and the sink are identical. Following this assessment, it confirms whether the source's current surpasses that of the sink. Subsequently, it formulates the RDO values by deriving insights from this evaluation.

Please note that certain parts of this text have been intentionally masked or hidden for confidentiality reasons. This is to protect sensitive information from unauthorized access or disclosure. Thank you for your understanding

Figure 7: VAR2FIX loop

As we can see in the figure 7 the algorithm assesses if the value of the fixed PDO from the source falls within the boundaries set by the source's minimum and maximum voltage limits. Moreover, it verifies whether the current value of the fixed PDO from the source is greater or lesser than the maximum current of the sink. Subsequently, it constructs the RDO considering these factors.



Please note that certain parts of this text have been intentionally masked or hidden for confidentiality reasons. This is to protect sensitive information from unauthorized access or disclosure. Thank you for your understanding

Figure 8: VAR2AVS loop

For the AVS algorithm the only bit fields that are used are the 4 MSB [1] in case the PDO is a AV SPR APDO it generates the values of avs_voltage and avs_current following this rules:

[1]Most Significant Bits

avs_voltage:

- voltage in the fixerd source PDO with the highest index and non-null current

avs_current:

- if Output_v $>$ 15V OR avs_voltage $<$ 15.1V
  - avs_current = the current in the fixed source PDO with the highest index and non-null current
- Else
  - avs_current = the current in the fixed source PDO with the second highest index.

As we can see in the figure 8 the algorithm evaluates whether the avs_voltage value surpasses the sink's minimum threshold and if the avs_voltage is greater than 9V and if the sink's maximum voltage also exceeds 9V. Subsequent to these assessments, it determines whether the avs_current is lower or higher than the maximum current, and then proceeds to formulate the RDO in accordance with this criteria.

Please note that certain parts of this text have been intentionally masked or hidden for confidentiality reasons. This is to protect sensitive information from unauthorized access or disclosure. Thank you for your understanding

Figure 9: FIX2AVS loop

As we can see in the figure 9 the algorithm assesses if the avs_voltage is greater than or equal to the fixed voltage of the sink, and also checks if the sink's fixed voltage is greater than or equal to 9V, in addition to verifying if the avs_voltage exceeds 9V. Following this, it evaluates whether the avs_current is greater or smaller than the current value of the sink's fixed PDO, the RDO is then formulated based on this evaluation.

Three distinct categories of RDOs are produced through the matching process, with a focus on maximizing power. In situations where multiple matches yield identical power levels, preference is given to the match characterized by the greatest voltage, which corresponds to the lowest current. These three types include:

FIXED RDO - Used in the FIX2FIX and VAR2FIX

0	B31 Reserved
obj_pos[2:0]	B30...28 Object position (0000b and 1110b...1111b are Reserved and Shall not be used)
0	B27 GiveBack flag
mismatch	B26 Capability Mismatch
USB_COM_CAPABLE(REG)	B25 USB communications Capable
1	B24 NO USB Suspend, initiated to 1 but can be update by MCU in the TX buffer
0	B23 Unchunked Extended Messages Supported, initiated to 0 but can be updated by MCU in the TX buffer
EPR_CAPABLE(REG)	B22 EPR Mode Capable
0	B21..20 Reserved - Shall be set to zero
op_curr[9:0]	B19...10 Operating Current in 10mA units
max_curr[9:0]	B9...0 Max Current in 10mA units

Table 1: FIXED RDO

The power requested by the FIX RDO is obtained by multiplying the op_curr[9:0] with the voltage match. The voltage match corresponds to the SRC FIX PDO situated at the object's position within bits [19:10].

PPS RDO - Used in the PPS2PPS

0	B31 Reserved
obj_pos[2:0]	B30...28 Object position (0000b and 1110b...1111b are Reserved and Shall not be used)
0	B27 GiveBack flag
mismatch	B26 Capability Mismatch
USB_COM_CAPABLE(REG)	B25 USB communications Capable
1	B24 NO USB Suspend, initiated to 1 but can be update by MCU in the TX buffer
0	B23 Unchunked Extended Messages Supported, initiated to 0 but can be updated by MCU in the TX buffer
EPR_CAPABLE(REG)	B22 EPR Mode Capable
0	B21 Reserved - Shall be set to zero
output_v[11:0]	B20...9 Output Voltage in 20mV units
0	B8...7 Reserved - Shall be set to zero
op_curr[6:0]	B6...0 Operating Current in 50mA units

Table 2: PPS RDO

The power requested by the PPS RDO is obtained by multiplying the op_curr[6:0] and the output_v[11:0].

AVS RDO - Used in the VAR2AVS and FIX2AVS

0	B31 Reserved
obj_pos[2:0]	B30...28 Object position (0000b and 1110b...1111b are Reserved and Shall not be used)
0	B27 GiveBack flag
mismatch	B26 Capability Mismatch
USB_COM_CAPABLE(REG)	B25 USB communications Capable
1	B24 NO USB Suspend, initiated to 1 but can be update by MCU in the TX buffer
0	B23 Unchunked Extended Messages Supported, initiated to 0 but can be updated by MCU in the TX buffer
EPR_CAPABLE(REG)	B22 EPR Mode Capable
0	B21 Reserved - Shall be set to zero
output_v[11:0]	B20...9 Output Voltage in 25mV units the least two significant bits are set to zero making the effective voltage step size 100mV
0	B8...7 Reserved - Shall be set to zero
op_curr[6:0]	B6...0 Operating Current in 50mA units

Table 3: AVS RDO

the power requested by the AVS RDO is obtained by multiplying the op_curr[6:0] and the output_v[11:0].

5 The Lazify Tool

Throughout the creation process of the Lazify tool, numerous discussions took place within the team, as well as with other verification teams in ST, and there were also consultations with the university tutor regarding the appropriate format for describing the state machine.

Utilizing the tool is facilitated by a comprehensive help command, accessible to users, which furnishes vital information for its effective operation. Moreover, a README file is included upon cloning the associated Git repository, offering essential insights into the tool's setup and usage.

Furthermore, a user manual serves as an in-depth guide, delving into the intricacies of the code, accompanied by extensive comments. This trifecta of resources ensures that users are well-equipped to harness the tool's capabilities, catering to a spectrum of information needs and learning preferences.

5.1 First Step - Format Specification

When engaged in the design of diverse circuits, designers frequently encounter a range of choices in terms of expressing specifications through various formats. Addressing this well-known challenge, the verification community consistently strives to find solutions. Here are a few instances of the formats most frequently employed:

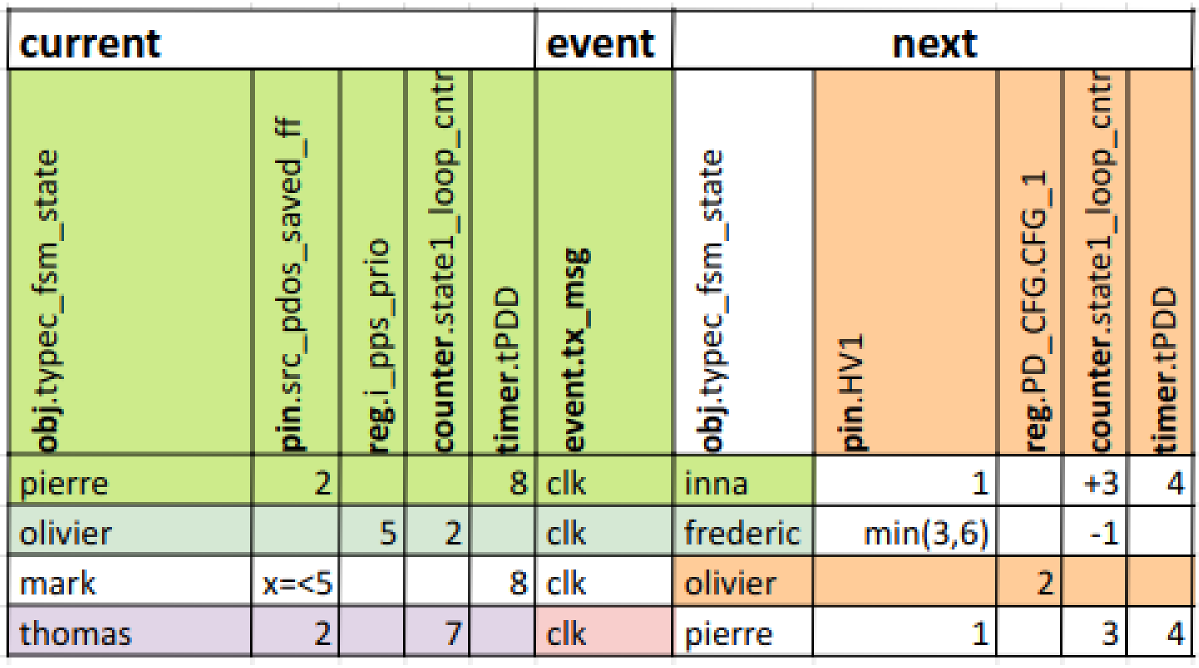
- Text.
- Visual State or UML Diagram.
- Excel table.
- A mix off all the above.

In this undertaking, the goal was to address this concern too. Acknowledging the significance of creating a uniform structure was deemed crucial, one that could function as an input for the tool. Furthermore, the intention was for this arrangement to provide a concise portrayal of the VPlan.

Not just that, this arrangement ought to also be:

- Human readable: It should be easily understandable and interpretable by a human being without the need for specialized knowledge or tools.
- Universal: It should be applicable, relevant, or existing everywhere or to everyone.
- Machine Extractable: The data or information should be easily and accurately extracted by computer programs or algorithms.

Two primary formats were considered: UML diagrams and XLS sheets. The project ultimately opted for the latter, driven by the fact that it is a format already employed by professors to educate future engineers. Additionally, this choice encompassed all the required features for the project's implementation.



current					event	next				
obj.typec_fsm_state	pin.src_pdos_saved_ff	reg.i_pps_prio	counter.state1_loop_cntr	timer.tPDD	event.tx_msg	obj.typec_fsm_state	pin.HV1	reg.PD_CFG.CFG_1	counter.state1_loop_cntr	timer.tPDD
pierre	2			8	clk	inna	1		+3	4
olivier		5	2		clk	frederic	min(3,6)		-1	
mark	x=<5			8	clk	olivier		2		
thomas	2		7		clk	pierre	1		3	4

Figure 10: Format for the Lazify Tool

The decision was made to partition the Excel table into three distinct sections: the **current** state, the triggering **event** for the transition, and the **next** state. To initiate the transition, all conditions of the current state must be satisfied. If a value is absent, it signifies the absence of validation for that particular variable.

For example, in case the object **typec_fsm_state** is pierre **AND** the pin **src_pdos _saved_ff** is 2 **AND** the timer **tPDD** is 8 the transition is activated and the object **typec_fsm_state** becames inna **AND** the pin HV1 1 **AND** the counter **PD_CFG .CFG_1** is increased by 3 **AND** the timer **tPDD** is 4.

As depicted in figure 10, numerous possibilities exist for the values, encompassing a spectrum from text and expressions to variable increments, decrements, and even value intervals.

As it is possible to see from the example there are a few keys words to describe the digital system, there is:

- Pins -> pin : Physical Pins
- Timers -> timer : Physical Timers
- Counters -> counter: Physical Counter (Has the possibility to increase and decrease the value)
- Registers -> reg: Physical Register
- Events -> event: Event, for example clock
- Object -> obj: The object being delineated within the Testbench, albeit not in a physical sense, necessitates manual description and inscription within the testbench code.

These keywords will align with the conditions being used in the database, which will be elucidated in the subsequent section.

5.2 Second Step - Parser

The parser was constructed using Python, primarily because of its widespread popularity as a high-level language, and it is also increasingly gaining traction within the digital verification community.

Another crucial rationale for employing Python in this project is its inherent support for classes. This attribute provides the opportunity to create an object-oriented database structure, enabling a more organized and representative depiction of verification objects.

To interact with Excel, the utilization of libraries is essential. For this project, the selected Python library is *openpyxl*. This choice was made owing to several factors: its lightweight nature compared to alternative libraries, its open-source nature with active development, compatibility, the fact that it is implemented purely in Python, and its capability for document structure manipulation.

The core and pivotal aspect of the project is the database, serving as the repository for the information extracted by the parser from the Excel sheet. This database format is designed to be flexible, highly comprehensible, and equipped with the capacity to seamlessly accommodate additions or updates.

5.2.1 Data Base

The significance of adopting a language that facilitates object-oriented programming cannot be overstated, especially in the context of the database. Each condition possesses distinct attributes, yet they inherit from a common class. This approach greatly simplifies both programming and comprehension of the problem. Moreover, Python's ability to support multiple inheritance adds a crucial dimension to the project's implementation.

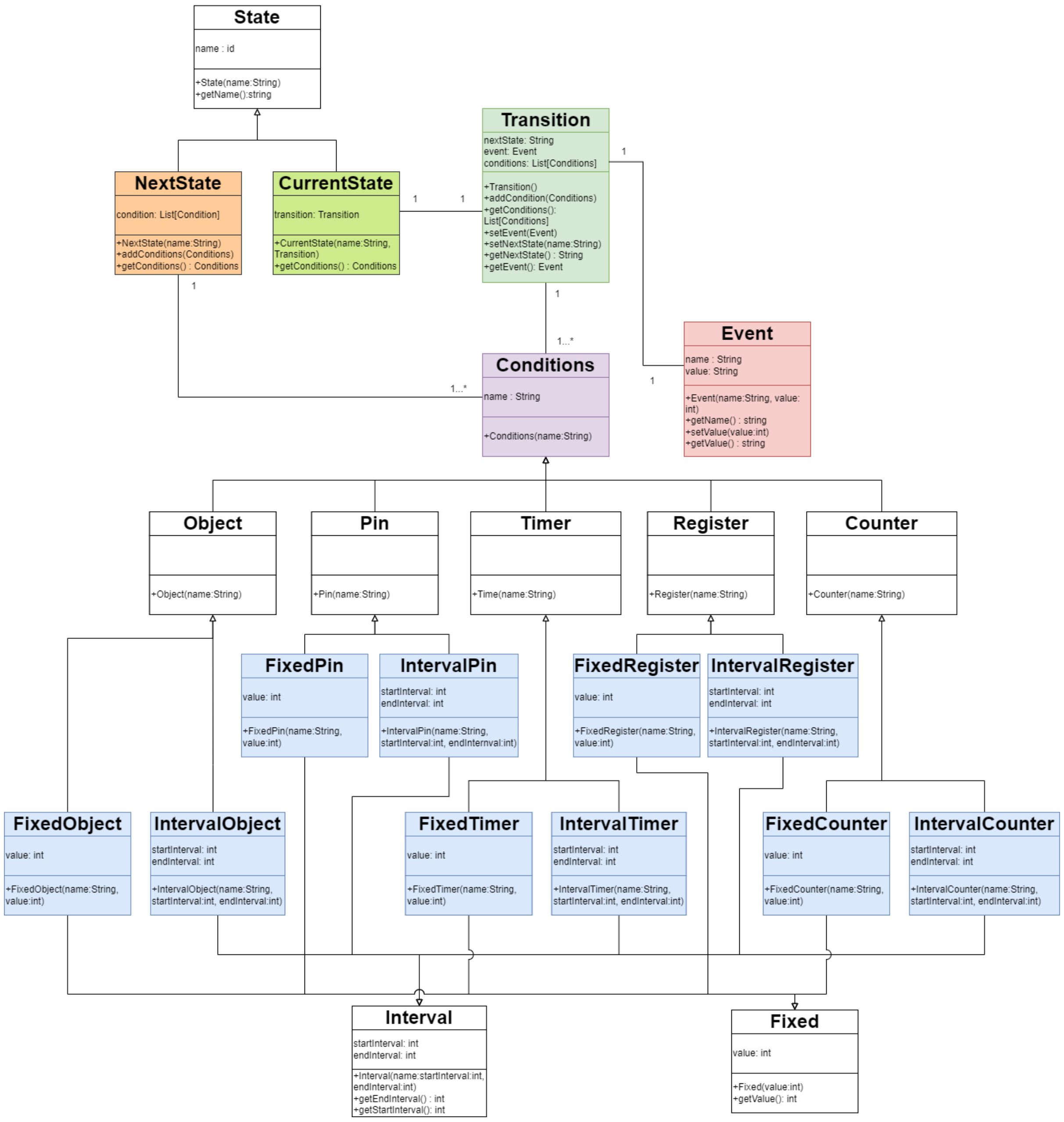


Figure 11: Data Base used in Lazify.

Fromm what is being showed in the figure 11 the main classes in the database are the NextState and the CurrentState, both behind extended from the State class.

The CurrentState is a class encompassing a single transition. Within this transition there is an NextState which represents the destine for the transition, an event that serves as the trigger, this trigger is activates the transition when all conditions are satisfied.

These conditions can take the form of Object, Pin, Timer, Register, or Counters. These conditions are initially conceptualized as abstract classes and attain concreteness only when they fall into the categories of Fixed or Interval. This distinction is vital due to the unique nature of Intervals, which entail two values and distinct handling for code automation.

On the other hand, the NextState does not involve a transition; it exclusively comprises conditions. As the destination state, it represents the endpoint. In this scenario, the conditions could be appropriately referred to as predictions or anticipated values for the NextState.

5.2.2 Parser

The parser operates by first identifying merged cells within the Excel sheet. This underscores the significance of adhering to the precise Excel sheet format as illustrated in figure 10. Once identified, the parser commences the construction of the database for the CurrentState, utilizing the conditions. It further reads the subsequent two cells to incorporate the event and the ensuing state.

With the CurrentState established, the parser proceeds to assemble the NextState, employing the boundaries established by the merged cells. The nomenclature of current and next of these merged cells bears considerable importance, as they serve as cues enabling the parser to accurately navigate its processing.

Furthermore, the significance of condition names cannot be overstated. If a condition is omitted, the parser will disregard it entirely, generating a database that does not reflect the intended description. This meticulous handling ensures that the generated database remains aligned with the precise description and intended representation of the system.

5.3 Third Step - Generating Code.

The third and pivotal step of the process entails the actual generation of code. This script is multifaceted, encompassing the reference model, as well as assertions. The purpose of this code is to leverage the data stored within the meticulously constructed database, which was meticulously crafted by the parser in the preceding steps. Through this code generation, the system harnesses the comprehensive information within the database to manifest a functional and coherent implementation.

The functioning mechanism involves the sequential extraction of the initial state from the CurrentState and the subsequent state from the NextState. Thereby culminating in a code structure resembling the exemplified code below:

```
if(typec_fsm_state==pierre && src_pdos_saved_ff==2 && tPDD==8) begin
    typec_fsm_state = inna;
    HV1 = 1;
    state1_loop_cntr+=3;
    tPDD = 4;
end
else if(typec_fsm_state == olivier && i_pps_prio == 5 &&
   state1_loop_cntr == 2) begin
    typec_fsm_state = frederic;
    HV1 = min(3,6);
    state1_loop_cntr-=1;
end
else if(typec_fsm_state == mark && src_pdos_saved_ff=<5 && tPDD == 8)
   begin
    typec_fsm_state = olivier;
    PD_CFG.CFG_1 = 2;
end
else if(typec_fsm_state == thomas && src_pdos_saved_ff == 2 &&
   state1_loop_cntr == 7) begin
    typec_fsm_state = pierre;
    HV1 = 1;
    state1_loop_cntr = 3;
    tPDD = 4;
end
```

It's important to note that the aforementioned code was generated based on the Excel format depicted in figure 10.

This approach ensures a methodical translation of the data into executable code, leading to an outcome that mirrors the description of the design made in the excel sheet.

Expanding the tool's capabilities necessitates the creation of distinct code segments for code generation, each tailored to the specific functionality. These segments would also entail the extraction of relevant information from the database. To integrate these new functionalities, a simple addition would involve introducing an argument to invoke the appropriate code segment within the main script. This modular approach allows seamless integration of new features while maintaining the tool's overall structure and usability.

6 Implementing the tool in the STUSB4531 project

The DPM was selected as the initial test vehicle for the tool due to its inherent complexity, as evident from the insights presented in section 4. This choice is driven by the presence of two concurrent state machines within the DPM, operating in parallel. Furthermore, the challenge arising from varying unit resolutions further accentuates the intricacy of the DPM, making it an ideal candidate to thoroughly evaluate the tool's capabilities and robustness.

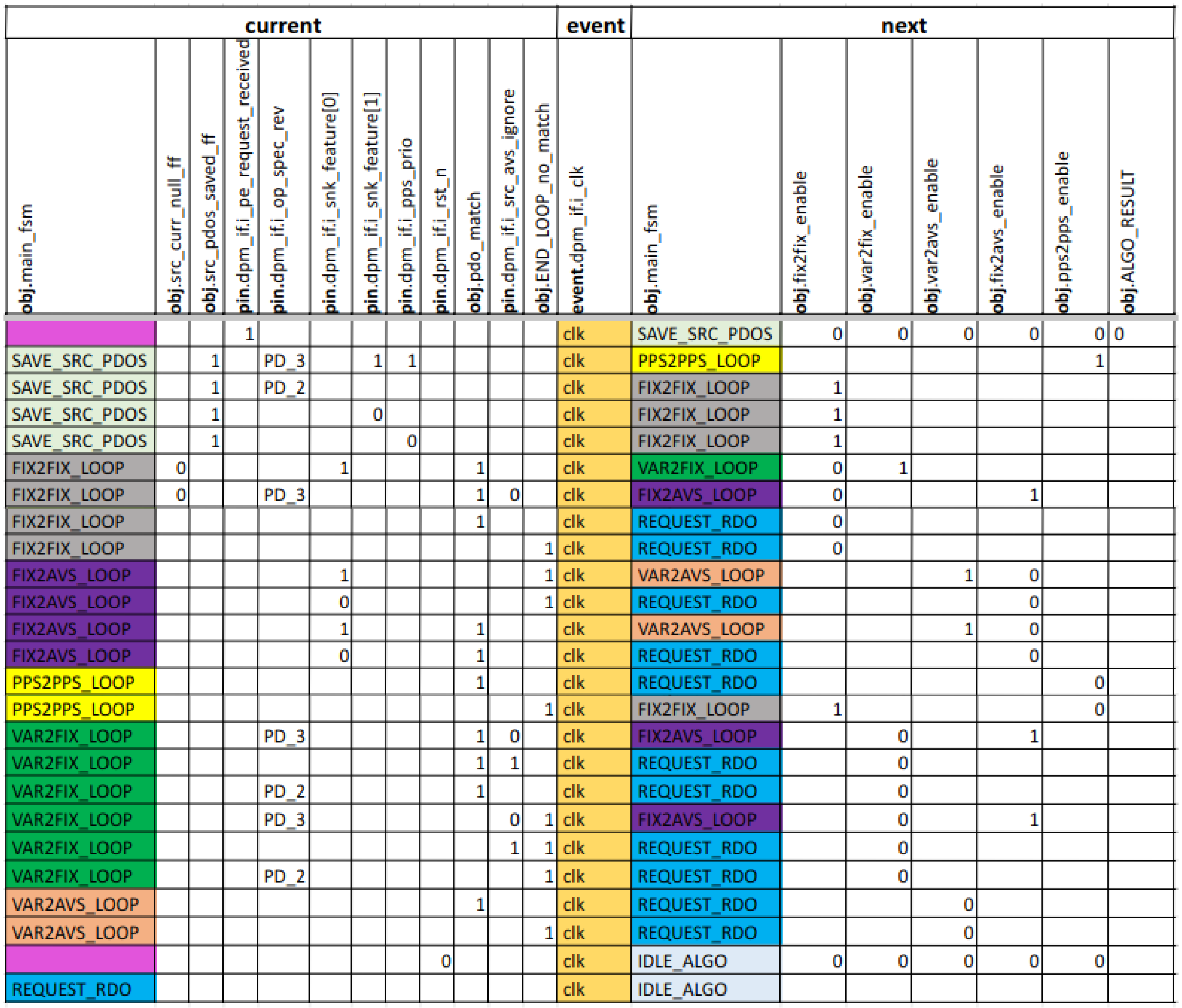
In addition to these reasons, it's worth noting that there was already a dedicated verification engineer engaged in the validation of the block. This assurance of a thorough verification process already in place contributed to our confidence that the circuit's validation would remain accurate even with the tool's development. This collaborative approach not only reinforced the tool's credibility but also ensured a reliable and comprehensive verification outcome for the circuit.

Given the ongoing nature of the design specification and the relatively less familiar domain, extensive interactions were essential with the design team. These discussions aimed to comprehensively grasp the functioning and intricacies of the block. This collaborative effort was imperative to ensure a clear understanding of the block's intended operation and to align the tool's development with the project's requirements. The iterative dialogue fostered effective communication, enabling the tool to accurately cater to the unique aspects of the subject matter.

The main focus during the initial test phase revolved around creating a reference model, commonly known as a scoreboard, designed specifically to cater to the intricacies of the DPM. The objective was to create a benchmark for comparison, facilitating the evaluation of the autogenerate code's outcomes against the actual results produced by the circuit. This approach enabled a comprehensive assessment of the tool's accuracy and effectiveness, ensuring a thorough validation of the circuit's functionality through a robust and systematic comparison process.

Indeed, this endeavor went beyond mere comparison. It acted as a method to validate and comprehensively assess the tool's functionalities. By subjecting the autogenerated code's results to comparison with the actual circuit's outcomes, not only were potential discrepancies highlighted, but also the tool's strengths and weaknesses were illuminated. This iterative validation process not only bolstered confidence in the tool's reliability but also provided invaluable insights into its performance nuances, paving the way for refinement and enhancement.

To facilitate this comparison, a rudimentary checker was devised to verify the values of register against the value generated by the tool.



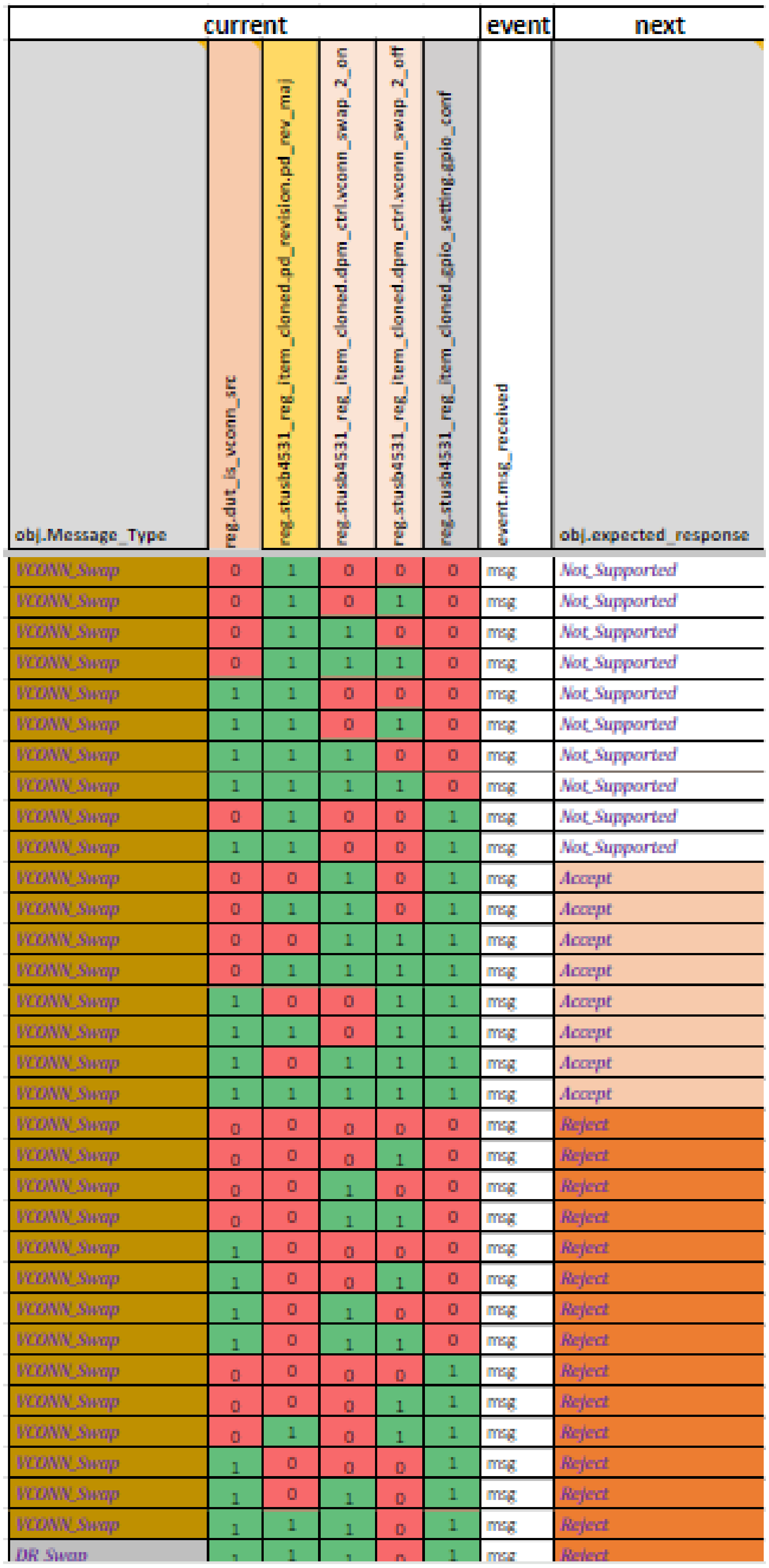
current												event	next						
obj.main_fsm	obj.src_curr_null_ff	obj.src_pdos_saved_ff	pin.dpm_if.i_pe_request_received	pin.dpm_if.i_op_spec_rev	pin.dpm_if.i_snk_feature[0]	pin.dpm_if.i_snk_feature[1]	pin.dpm_if.i_pps_prio	pin.dpm_if.i_rst_n	obj.pdo_match	pin.dpm_if.i_src_avs_ignore	obj.END_LOOP_no_match	event.dpm_if.i_clk	obj.main_fsm	obj.fix2fix_enable	obj.var2fix_enable	obj.var2avs_enable	obj.fix2avs_enable	obj.pps2pps_enable	obj.ALGO_RESULT
			1									clk	SAVE_SRC_PDOS	0	0	0	0	0	0
SAVE_SRC_PDOS		1		PD_3		1	1					clk	PPS2PPS_LOOP					1	
SAVE_SRC_PDOS		1		PD_2								clk	FIX2FIX_LOOP	1					
SAVE_SRC_PDOS		1				0						clk	FIX2FIX_LOOP	1					
SAVE_SRC_PDOS		1					0					clk	FIX2FIX_LOOP	1					
FIX2FIX_LOOP	0				1				1			clk	VAR2FIX_LOOP	0	1				
FIX2FIX_LOOP	0			PD_3					1	0		clk	FIX2AVS_LOOP	0			1		
FIX2FIX_LOOP									1			clk	REQUEST_RDO	0					
FIX2FIX_LOOP											1	clk	REQUEST_RDO	0					
FIX2AVS_LOOP					1						1	clk	VAR2AVS_LOOP			1	0		
FIX2AVS_LOOP					0						1	clk	REQUEST_RDO				0		
FIX2AVS_LOOP					1				1			clk	VAR2AVS_LOOP			1	0		
FIX2AVS_LOOP					0				1			clk	REQUEST_RDO				0		
PPS2PPS_LOOP									1			clk	REQUEST_RDO					0	
PPS2PPS_LOOP											1	clk	FIX2FIX_LOOP	1				0	
VAR2FIX_LOOP				PD_3					1	0		clk	FIX2AVS_LOOP		0		1		
VAR2FIX_LOOP									1	1		clk	REQUEST_RDO		0				
VAR2FIX_LOOP				PD_2					1			clk	REQUEST_RDO		0				
VAR2FIX_LOOP				PD_3						0	1	clk	FIX2AVS_LOOP		0		1		
VAR2FIX_LOOP										1	1	clk	REQUEST_RDO		0				
VAR2FIX_LOOP				PD_2							1	clk	REQUEST_RDO		0				
VAR2AVS_LOOP									1			clk	REQUEST_RDO			0			
VAR2AVS_LOOP											1	clk	REQUEST_RDO			0			
								0				clk	IDLE_ALGO	0	0	0	0	0	
REQUEST_RDO												clk	IDLE_ALGO						

Figure 12: Main Loop Excel Sheet

The figure 12 illustrates the Excel representation of the main loop, as depicted in figure 4. And the code in the annex 8.1

The tool's value became evident as it revealed previously undetected bugs, despite the concurrent efforts of another verification engineer on the block. Additionally, the tool's comprehensive verification process resonated well with the design team, as it not only identified existing issues but also offered a rigorous level of verification that garnered their appreciation. This dual impact showcased the tool's ability to enhance verification depth and accuracy, garnering positive feedback from both the verification and design stakeholders.

Subsequent to the completion of the reference model generation, another test vehicle was proposed for the project. This involved the generation of assertions to validate message responses for the Policy Engine (PE). These assertions were designed to verify the expected responses based on the configuration of the Device Under Testing (DUT). This extension of the tool's functionality aimed to ensure comprehensive protocol compliance testing, encompassing various DUT configurations and corresponding response scenarios.



current						event	next
obj.Message_Type	reg.dut_is_vconn_src	reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj	reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on	reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off	reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf	event.msg_received	obj.expected_response
VCONN_Swap	0	1	0	0	0	msg	Not_Supported
VCONN_Swap	0	1	0	1	0	msg	Not_Supported
VCONN_Swap	0	1	1	0	0	msg	Not_Supported
VCONN_Swap	0	1	1	1	0	msg	Not_Supported
VCONN_Swap	1	1	0	0	0	msg	Not_Supported
VCONN_Swap	1	1	0	1	0	msg	Not_Supported
VCONN_Swap	1	1	1	0	0	msg	Not_Supported
VCONN_Swap	1	1	1	1	0	msg	Not_Supported
VCONN_Swap	0	1	0	0	1	msg	Not_Supported
VCONN_Swap	1	1	0	0	1	msg	Not_Supported
VCONN_Swap	0	0	1	0	1	msg	Accept
VCONN_Swap	0	1	1	0	1	msg	Accept
VCONN_Swap	0	0	1	1	1	msg	Accept
VCONN_Swap	0	1	1	1	1	msg	Accept
VCONN_Swap	1	0	0	1	1	msg	Accept
VCONN_Swap	1	1	0	1	1	msg	Accept
VCONN_Swap	1	0	1	1	1	msg	Accept
VCONN_Swap	1	1	1	1	1	msg	Accept
VCONN_Swap	0	0	0	0	0	msg	Reject
VCONN_Swap	0	0	0	1	0	msg	Reject
VCONN_Swap	0	0	1	0	0	msg	Reject
VCONN_Swap	0	0	1	1	0	msg	Reject
VCONN_Swap	1	0	0	0	0	msg	Reject
VCONN_Swap	1	0	0	1	0	msg	Reject
VCONN_Swap	1	0	1	0	0	msg	Reject
VCONN_Swap	1	0	1	1	0	msg	Reject
VCONN_Swap	0	0	0	0	1	msg	Reject
VCONN_Swap	0	0	0	1	1	msg	Reject
VCONN_Swap	0	1	0	1	1	msg	Reject
VCONN_Swap	1	0	0	0	1	msg	Reject
VCONN_Swap	1	0	1	0	1	msg	Reject
VCONN_Swap	1	1	1	0	1	msg	Reject
DR_Swap	1	1	1	0	1	msg	Reject

Figure 13: Assertions Excel Sheet

As evidenced in figure 13, the table structure for the assertions closely resembles that of the reference model, as it adheres to the same format.

The underlying concept remains consistent with that of the reference model. The identical parser is employed, with the differentiating factor being the utilization of an alternate script tailored for generating assertions. This approach ensures continuity in terms of methodology while accommodating the specific requirements for generating assertions pertaining to message responses.

The outcome of the assertions can be observed in annex 8.2.

Examining the assertions sheet, it becomes evident that several lines could potentially be streamlined for response validation. Contemplating this, a proposal for implementing boolean reduction of the table emerges as a viable solution.

By leveraging an open-source code from GitHub [2] and making slight adaptations, the implementation process was streamlined. The end result is the assertion code, which has been simplified through boolean reduction. This reduced assertion code can be examined in Annex 8.3.

Upon comparing the two sets of code, a substantial disparity in the number of lines and the enhanced readability is apparent. However, it's important to acknowledge that this streamlined approach comes at the cost of losing specific information, particularly the ability to discern which line of assertions triggered a particular assertion.

Discussions also arose about incorporating a new feature into Lazify, specifically "equivalence checking." However, the introduction of this feature encountered a roadblock due to a disparity in the design partition, particularly the pin-out, between the RTL and the reference model generated by Lazify.

7 Conclusion

The culmination of this project marks a significant stride forward in the realm of verification, underscoring the effectiveness of modern tooling in ensuring the robustness and reliability of complex digital systems. The journey embarked upon revolved around the development of a versatile and potent tool capable of systematically generating reference models, crafting assertions, and streamlining verification procedures. The motivation driving this effort stemmed from the acknowledgment of the increasing complexities inherent in modern digital designs. This recognition prompted the need for inventive solutions to effectively address the challenges associated with thorough verification.

From its inception, the project set its sights on an ambitious objective: to design a tool that could seamlessly integrate with existing design processes, thereby bolstering the efficacy of verification methodologies. The selection of Python as the implementation language was a strategic choice, capitalizing on its object-oriented nature and widespread usage within the digital verification community. This choice paved the way for a multi-step approach, beginning with the construction of an intricate database through the parsing of Excel sheets. This meticulous extraction and organization of data formed the base for subsequent processes.

The first test vehicle, focusing on the generation of a reference model, demonstrated the tool's capability to uncover subtle bugs that had gone unnoticed in previous assessments. Notably, the ability to validate against an established verification engineer's efforts underscored the tool's added value. Moreover, the tool's capacity to harmonize with diverse design specifications, guided by meticulous discussions with the design team, further attested to its adaptability.

The exploration didn't stop there. The tool's prowess was extended to assertions for message responses, thus broadening its utility to encompass a spectrum of verification needs. This expansion aligned with the intricate nature of digital designs, where customized assertions play a crucial role in ensuring the correct operation of various configurations. The implementation of boolean reduction further underscored the tool's adaptability, resulting in more concise assertion code while acknowledging the trade-off of lost granularity.

Looking back,, the journey embarked upon was a comprehensive exploration of modern verification challenges, yielding a multifaceted tool that empowers verification engineers to navigate the complexities of digital design with enhanced efficiency and rigor. This project's success is a testament to the collaborative synergy between technology and human expertise, signaling a promising path forward in the ever-evolving landscape of digital verification.

7.1 Current Limitations and Future Prospects

While the developed tool marks a substantial leap forward in automating verification processes, it is important to acknowledge certain current limitations. One notable aspect is the extent of manual code writing that persists, particularly in cases such as defining testbench signals. At present, setting up these signals requires a significant investment of time and effort, especially during the initial stages of implementation.

Despite this limitation, it is worth highlighting that the tool's true potential unfolds in the long run. The up-front time investment to meticulously establish the required signals and configurations gradually yields significant dividends as the verification process progresses, which is a common occurrence during development, given that verification often commences while the design itself is still undergoing creation and modifications. This dynamic process between design evolution and ongoing verification necessitates a flexible approach that acknowledges the need for iterative adjustments and enhancements to the tool's capabilities.

This inherent trade-off, where an initial manual effort translates into substantial time savings over extended verification cycles, is a hallmark of the tool's pragmatic value.

As the tool continues to evolve, there exists a promising trajectory for further automation, potentially encompassing areas that currently necessitate manual intervention.

By addressing these aspects, the tool's efficiency and effectiveness could be further increased, streamlining the overall verification process even more comprehensively. The ongoing pursuit of refining and augmenting the tool holds the promise of bridging the current gap between manual involvement and automated verification, this sets the stage for a period marked by increased efficiency and greater reliability in the verification of digital designs.

In conclusion, while the current state of the tool acknowledges certain manual aspects, its ability to revolutionize the verification landscape remains indisputable. As advancements are made and the tool matures, the balance between initial investment and long-term gains will undoubtedly shift further in favor of enhanced efficiency and expedited verification processes.

7.2 Impact

During the course of the internship, a pivotal moment transpired through a collaborative meeting with the verification teams both at ST Grenoble and on a global scale across ST. This event served as a platform for presenting the project's progress and intricacies, having notable interest from several teams. The impact of the tool's abilities was evident during the discussions, as various teams showed excitement about incorporating the tool into their verification processes.

A particularly promising development emerged within the immediate working context, as the team directly engaged with during the internship is now actively being encouraged to initiate utilization of the tool. This growing interest within the company underscores the tool's practical usefulness and fits well with the broader aim of encouraging thorough verification practices throughout the organization.

The recognition and interest exhibited by the broader verification community, coupled with the internal impetus to adopt the tool, underscores its intrinsic value and its potential to drive positive transformation in verification methodologies. This recognition, fueled by genuine interest and practical application, serves as a testament to the tool's significance and the impact it stands to make in streamlining and enhancing the verification process.

7.3 Personal Conclusion

The internship journey has been truly fulfilling, offering a chance to make meaningful strides in the project. The work environment was fantastic, with supportive and friendly colleagues who were always ready to assist. The possibility of future collaboration with ST was also extended, which was a great honor. Although I had to defer this opportunity due to my return to Brazil, ongoing discussions indicate potential for future engagement.

This internship experience has been incredibly valuable, not just for the technical accomplishments but also for the professional relationships and opportunities that have emerged. The impact of this experience is felt not only in what has been achieved, but also in the exciting prospects that may lie ahead. ST has proven to be a place where hard work and contributions are acknowledged and welcomed.

8 Annexes

8.1 Main Loop Code

```
if(dpm_if.i_pe_request_received == 1) begin
    main_fsm = SAVE_SRC_PDOS;
    fix2fix_enable = 0;
    var2fix_enable = 0;
    var2avs_enable = 0;
    fix2avs_enable = 0;
    pps2pps_enable = 0;
    ALGO_RESULT = 0;
end
else if(main_fsm == SAVE_SRC_PDOS && src_pdos_saved_ff == 1 && dpm_if.i_op_spec_rev == PD_3 && dpm_if.i_snk_feature[1] == 1 && dpm_if.i_pps_prio == 1) begin
    main_fsm = PPS2PPS_LOOP;
    pps2pps_enable = 1;
end
else if(main_fsm == SAVE_SRC_PDOS && src_pdos_saved_ff == 1 && dpm_if.i_op_spec_rev == PD_2) begin
    main_fsm = FIX2FIX_LOOP;
    fix2fix_enable = 1;
end
else if(main_fsm == SAVE_SRC_PDOS && src_pdos_saved_ff == 1 && dpm_if.i_snk_feature[1] == 0) begin
    main_fsm = FIX2FIX_LOOP;
    fix2fix_enable = 1;
end
else if(main_fsm == SAVE_SRC_PDOS && src_pdos_saved_ff == 1 && dpm_if.i_pps_prio == 0) begin
    main_fsm = FIX2FIX_LOOP;
    fix2fix_enable = 1;
end
else if(main_fsm == FIX2FIX_LOOP && src_curr_null_ff == 0 && dpm_if.i_snk_feature[0] == 1 && pdo_match == 1) begin
    main_fsm = VAR2FIX_LOOP;
    fix2fix_enable = 0;
    var2fix_enable = 1;
end
else if(main_fsm == FIX2FIX_LOOP && src_curr_null_ff == 0 && dpm_if.i_op_spec_rev == PD_3 && pdo_match == 1 && dpm_if.i_src_avs_ignore == 0) begin
    main_fsm = FIX2AVS_LOOP;
    fix2fix_enable = 0;
    fix2avs_enable = 1;
end
else if(main_fsm == FIX2FIX_LOOP && pdo_match == 1) begin
    main_fsm = REQUEST_RDO;
    fix2fix_enable = 0;
end
else if(main_fsm == FIX2FIX_LOOP && END_LOOP_no_match == 1) begin
    main_fsm = REQUEST_RDO;
    fix2fix_enable = 0;
```

```
end
else if(main_fsm == FIX2AVS_LOOP && dpm_if.i_snk_feature[0] == 1 &&
    END_LOOP_no_match == 1) begin
     main_fsm = VAR2AVS_LOOP;
     var2avs_enable = 1;
     fix2avs_enable = 0;
end
else if(main_fsm == FIX2AVS_LOOP && dpm_if.i_snk_feature[0] == 0 &&
    END_LOOP_no_match == 1) begin
     main_fsm = REQUEST_RDO;
     fix2avs_enable = 0;
end
else if(main_fsm == FIX2AVS_LOOP && dpm_if.i_snk_feature[0] == 1 &&
    pdo_match == 1) begin
     main_fsm = VAR2AVS_LOOP;
     var2avs_enable = 1;
     fix2avs_enable = 0;
end
else if(main_fsm == FIX2AVS_LOOP && dpm_if.i_snk_feature[0] == 0 &&
    pdo_match == 1) begin
     main_fsm = REQUEST_RDO;
     fix2avs_enable = 0;
end
else if(main_fsm == PPS2PPS_LOOP && pdo_match == 1) begin
     main_fsm = REQUEST_RDO;
     pps2pps_enable = 0;
end
else if(main_fsm == PPS2PPS_LOOP && END_LOOP_no_match == 1) begin
     main_fsm = FIX2FIX_LOOP;
     fix2fix_enable = 1;
     pps2pps_enable = 0;
end
else if(main_fsm == VAR2FIX_LOOP && dpm_if.i_op_spec_rev == PD_3 &&
    pdo_match == 1 && dpm_if.i_src_avs_ignore == 0) begin
     main_fsm = FIX2AVS_LOOP;
     var2fix_enable = 0;
     fix2avs_enable = 1;
end
else if(main_fsm == VAR2FIX_LOOP && pdo_match == 1 && dpm_if.
    i_src_avs_ignore == 1) begin
     main_fsm = REQUEST_RDO;
     var2fix_enable = 0;
end
else if(main_fsm == VAR2FIX_LOOP && dpm_if.i_op_spec_rev == PD_2 &&
    pdo_match == 1) begin
     main_fsm = REQUEST_RDO;
     var2fix_enable = 0;
end
else if(main_fsm == VAR2FIX_LOOP && dpm_if.i_op_spec_rev == PD_3 &&
    dpm_if.i_src_avs_ignore == 0 && END_LOOP_no_match == 1) begin
     main_fsm = FIX2AVS_LOOP;
     var2fix_enable = 0;
     fix2avs_enable = 1;
end
```

```
else if(main_fsm == VAR2FIX_LOOP && dpm_if.i_src_avs_ignore == 1 &&
    END_LOOP_no_match == 1) begin
     main_fsm = REQUEST_RDO;
     var2fix_enable = 0;
end
else if(main_fsm == VAR2FIX_LOOP && dpm_if.i_op_spec_rev == PD_2 &&
    END_LOOP_no_match == 1) begin
     main_fsm = REQUEST_RDO;
     var2fix_enable = 0;
end
else if(main_fsm == VAR2AVS_LOOP && pdo_match == 1) begin
     main_fsm = REQUEST_RDO;
     var2avs_enable = 0;
end
else if(main_fsm == VAR2AVS_LOOP && END_LOOP_no_match == 1) begin
     main_fsm = REQUEST_RDO;
     var2avs_enable = 0;
end
else if(dpm_if.i_rst_n == 0) begin
     main_fsm = IDLE_ALGO;
     fix2fix_enable = 0;
     var2fix_enable = 0;
     var2avs_enable = 0;
     fix2avs_enable = 0;
     pps2pps_enable = 0;
end
else if(main_fsm == REQUEST_RDO) begin
     main_fsm = IDLE_ALGO;
end
```

8.2 Assertions

```
if(tx_msg.control_msg_resp == USB_PD_VCONN_SWAP) begin
     if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg
    .stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.
    stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.
    stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.
    dut_is_vconn_src   == 0) begin
          asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_1 :
    assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
          else `uvm_error(get_type_name(), $sformatf("Wrong response for
    VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.
    control_msg_resp.name()))
     end
     else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0
    && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 &&
    reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.
    stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.
    dut_is_vconn_src   == 0) begin
          asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_2 :
    assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
```

```
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_3 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_4 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_5 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_6 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_7 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_8 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_9 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_10 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_accept_1 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
 end
 else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_accept_2 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
```

```
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_3 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_4 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_5 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_6 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_7 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src    == 1) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_accept_8 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src    == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_reject_1 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src    == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_reject_2 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src    == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_reject_3 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
      else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src    == 0) begin
      asrt_exp_control_msg_resp_to_vconn_swap_is_reject_4 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_reject_5 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_reject_6 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_reject_7 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_reject_8 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
     else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
   end
   else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 0) begin
     asrt_exp_control_msg_resp_to_vconn_swap_is_reject_9 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
```

```
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_10 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_11 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_12 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.dut_is_vconn_src   == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_13 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src   == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_14 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else ‘uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
end
if(tx_msg.control_msg_resp == USB_PD_DR_SWAP) begin
    if(reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.dut_is_vconn_src    == 1) begin
        asrt_exp_control_msg_resp_to_dr_swap_is_reject_1 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else ‘uvm_error(get_type_name(), $sformatf("Wrong response for DR_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
end
```

8.3 Reduced Assertions

```
if(tx_msg.control_msg_resp == USB_PD_VCONN_SWAP) begin
    if(reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_1 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
        else ‘uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src    == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_not_supported_2 : assert( rx_msg.control_msg_resp == USB_PD_NOT_SUPPORTED)
        else ‘uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Not_Supported, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src    == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_accept_1 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
        else ‘uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src    == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_accept_2 : assert(rx_msg.control_msg_resp == USB_PD_ACCEPT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Accept, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_1 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src   == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_2 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src   == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_3 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
    else if(reg.dut_is_vconn_src   == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 0 && reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 0) begin
        asrt_exp_control_msg_resp_to_vconn_swap_is_reject_4 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for VCONN_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
end
if(tx_msg.control_msg_resp == USB_PD_DR_SWAP) begin
    if(reg.dut_is_vconn_src   == 1 && reg.stusb4531_reg_item_cloned.pd_revision.pd_rev_maj == 0 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_on == 1 && reg.stusb4531_reg_item_cloned.dpm_ctrl.vconn_swap_2_off == 1 && reg.stusb4531_reg_item_cloned.gpio_setting.gpio_conf == 1) begin
        asrt_exp_control_msg_resp_to_dr_swap_is_reject_1 : assert(rx_msg.control_msg_resp == USB_PD_REJECT)
        else `uvm_error(get_type_name(), $sformatf("Wrong response for DR_SWAP message: expected = Reject, actual = %s", rx_msg.control_msg_resp.name()))
    end
end
```

References

[1] "Stmicroeletronics." https://st.com.

[2] S. Adhikari, "Quine mccluskey algorithm for minimizing logical expressions." https://github.com/int-main/Quine-McCluskey, 2018.