Universitatea Transilvania din Brașov

Universitatea Transilvania din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

Departamentul: Inginerie Electrică și Știința Calculatoarelor

Programul de studii: Electronică Aplicată

BĂJENESCU Dragoș-Ionuț

BRUMARIU Cosmin-Nicușor

DESAGĂ Sergiu-Gabriel

SOLOMON Vlad-George

BORDEAN Aurel-Robert

PROIECT DE VERIFICARE

Brașov, 2024

Departamentul: Inginerie Electrică și Știința Calculatoarelor
Programul de studii: Electronică Aplicată

BĂJENESCU Dragoș-Ionuț
BRUMARIU Cosmin-Nicușor
SOLOMON Vlad-George
DESAGĂ Sergiu-Gabriel
BORDEAN Aurel-Robert

Verificarea funcționalităților cuptorului electric

Brașov, 2024

FIȘA PROIECTULUI DE VERIFICARE

<table>
<tr><td>Universitatea Transilvania din Braşov</td><td rowspan="2">Anul universitar:
2023-2024</td></tr>
<tr><td>Facultatea de Inginerie Electrică şi Ştiința Calculatoarelor</td></tr>
<tr><td>Membrii:
Băjenescu Dragoș-Ionuț, arhitect
Brumariu Cosmin-Nicușor, proiectant
Desagă Sergiu-Gabriel, inginer de verificare
Solomon Vlad-George, inginer de verfiicare
Bordean Aurel-Robert, inginer de verificare

Coordonatori:
Alexandru Dinu</td><td>
Sursa imaginii:
https://www.deltastudio.ro/cuptor-electric-fsm-86-h-bk-nero</td></tr>
<tr><td colspan="2">Titlul proiectului: Verificarea funcționalităților cuptorului electric</td></tr>
<tr><td colspan="2">Sumarul funcționalităților DUT-ului:
1. Funcția principală a DUT-ului este de a selecta modul de preparare, temperatura și timpul de preparare pentru o mâncare;
2. Să respecte funcționalitățile și timpul setat, să nu prezinte erori ce ar putea perturba funcționarea sistemului
3. Să fie capabil să preia datele pe care utilizatorul le dă prin intermediul interfeței APB</td></tr>
<tr><td colspan="2">Sumarul scenariilor de verificare:
1. Verifica daca led-ul se aprinde corect
2. Verificarea situatiei de la finalul operatiunii: daca dupa ce expira timpul se aprinde led-ul pentru confirmare
3. Daca usa se inchide si deschide la cererea noastra;</td></tr>
</table>

Istoricul versiunilor (se va completa pe tot parcursul semestrului, atunci când se va modifica specificația:

Ver-siune	Data finalizării	Numele editorului	Motiv
0.0	20.03.2021	Alexandru Dinu	Prima versiune de model

0.1	28.03.2022	Alexandru Dinu	Template-ul a fost actualizat
0.1	11.03.202	Brumariu Cosmin	S-a realizat schema DUT-ului
1.0	2.04.2024	Brumariu Cosmin	S-a realizat prima versiune a DUT-ului
1.0	9.04.2024	Brumariu Cosmin	S-au realizat specificațiile principale
1.1	27.04.2024	Bajenescu Dragos-Io nut	S-a trecut schema cu mediul de verificare
1.2	28.04.2024	Brumariu Cosmin	S-a adăugat interfața APB la DUT
1.2	02.05.2024	Bajenescu Dragos-Io nut	S-au trecut asertiunile in tabelul 2.2
1.3	23.04.2024	Desaga Sergiu Gabriel	S-a realizat fișierul de tranzactie pentru interfața APB
1.4	24.04.2024	Desaga Sergiu Gabriel	S-a realizat driverul pentru interfața APB
1.5	24.04.2024	Desaga Sergiu Gabriel	S-a realizat monitorul pentru interfața APB
1.6	24.04.2024	Desaga Sergiu Gabriel	S-a realizat generatorul pentru interfața APB
1.7	24.04.2024	Desaga Sergiu Gabriel	S-a realizat write and read in apb_transaction
1.8	24.04.2024	Desaga Sergiu Gabriel	S-a realizat coverage ul pentru interfața APB
1.9	24.04.2024	Desaga Sergiu Gabriel	S-a realizat integrarea tuturor fisierelor intr-un singur proiect
2.0	24.04.2024	Desaga Sergiu Gabriel	S-a modificat fișierul environment conform proiectului

2.1	24.04.2024	Desaga Sergiu Gabriel	S-a creat primul test funcțional.
2.2	24.04.2024	Desaga Sergiu Gabriel	S-au realizat teste pentru modurile de operare, temperatura și timp
2.3	07.05.2024	Desaga Sergiu Gabriel	S-a realizat testul pentru regiștrii
2.4	07.05.2024	Desaga Sergiu Gabriel	S-a realizat test pentru un mod de operare si timer
3.0	09.05.2024	Bajenescu Dragos-Ionut	Am modificat schema cu interfetele DUT-ului si am sters semnalele din interfata ready, in afara de semnalul de iesire + un desen mai sugestiv pentru mediul de verificare
3.1	10.05.2024	Bajenescu Dragos-Ionut	Am adaugat desenele pentru semnalele protocolului APB
3.2	10.05.2024	Brumariu Cosmin & Desaga Sergiu	Realizare teste modul timer , mod_operare , temperatura si fixare bug mod_ready

Cuprins

1 The digital design of the circuit.

1.1 Block Diagram

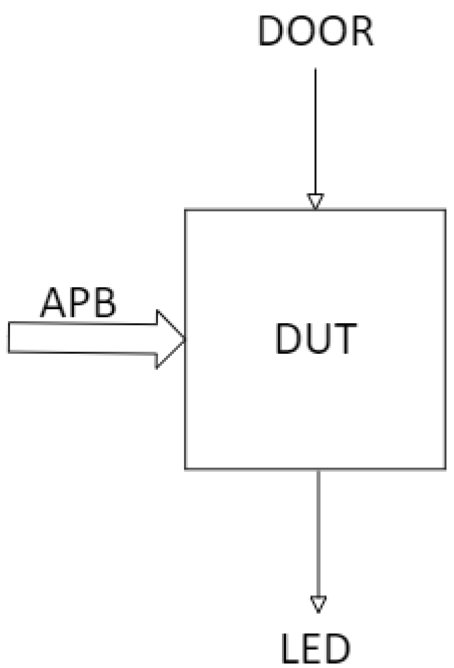

Figure 1. Interfaces used for the project

1.2 Description of the general functionality of the DUT (Device Under Test)

For this project, we chose to test the functionality of an electric oven. Initially, we selected five key functions of the oven: temperature settings, a timer for tracking the duration of each function (which will be set by the user), and a LED that will turn on when the current task is done (when the timer reaches zero, the LED indicates that the cooking process is complete and the dish can be removed from the oven). The LED light is up only if the door is closed. When the door opens, the led turns off. Also, the LED is not turned on in the IDLE state.

We will now describe the functionality of our project: A person has an electric oven at their disposal and can use five main functions to prepare food:

1. **Heating**: For simple reheating of dishes.
2. **Cooking**: For standard cooking processes.
3. **Defrosting**: Specifically for defrosting foods such as vegetables and fruits.
4. **Pizza Function**: Dedicated to baking pizza.

After selecting the desired mode, the temperature needs to be set. Depending on the user's preference, the oven offers temperature options ranging from 0° to 200°, in increments of 50° (0°, 50°, 100°, 150°, 200°). The user then sets a timer to define the cooking duration (the timer is initially set to 0). There is a common register (time_reg) which specifies the duration of the processes performed by the oven (heating, cooking, defrosting, pizza). All processes have the same duration.

Important: The reset function is active on 1.

The oven is equipped with a sensor that monitors the door. It will not begin operation until the door is fully closed, even if all other parameters have been set. Once the timer reaches zero, it signals that the cooking process is complete. This is indicated by the illumination of a red LED (the "ready" module), and optionally, a buzzer can sound to provide an audible alert in case the user is not near the oven.

If the user wishes to stop the cooking process before the timer runs out, they need to reset both the functionality and temperature settings to their initial state.

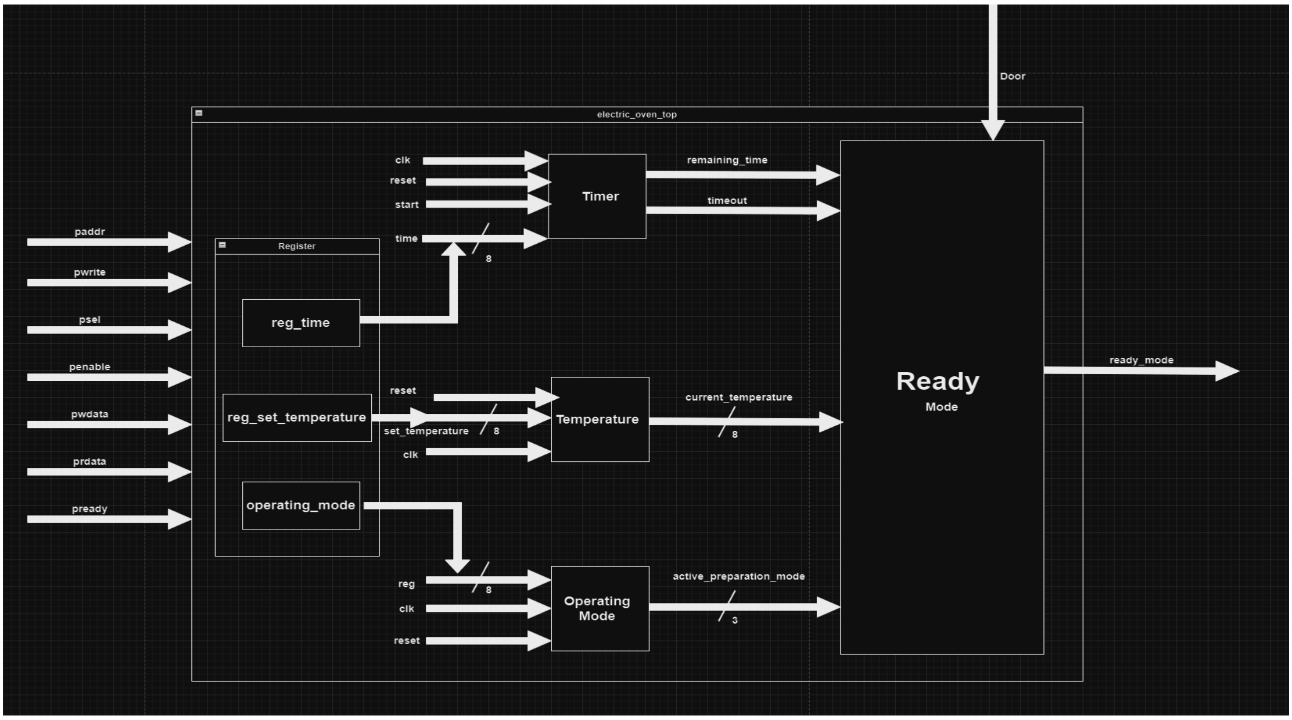

Figure 2. DUT Schematic

1.3 DUT Interfaces

Table 1. Signals in the data transmission interface

Interfața APB			
Signal Name	**Number of Bits**	**Direction**	**Description**
PCLK	1	Input	Clock signal
PRESETn	1	Input	Clock signal
PADDR	ADDR_ WIDTH	Output	Address signal, up to 32 bits.
PSELx	1	Output	Used for slave selection.
PENABLE	1	Output	Indicates the second and subsequent transfer cycles.

PWRITE	1	Output	Write access
PWDATA	DATA_WIDTH	Output	Write data bus when PWRITE is HIGH
PREADY	1	Input	The slave can accept data and address transfers
PRDATA	DATA_WIDTH	Input	Read data bus when PWRITE is LOW
PSLVERR(nu este folosit in cadrul proiectului)	1	Input	Optional signal in case of transfer errors

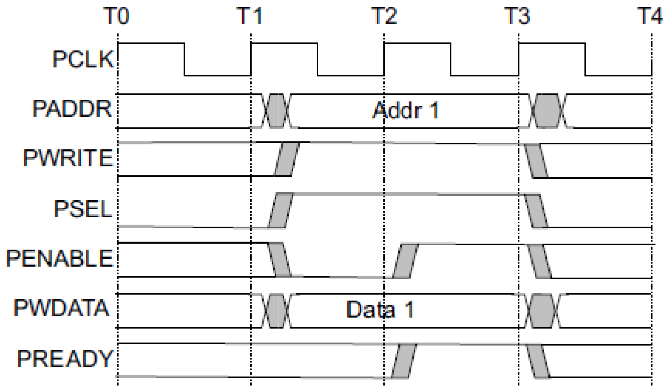

Fig. 3: APB Protocol Signals for Writing

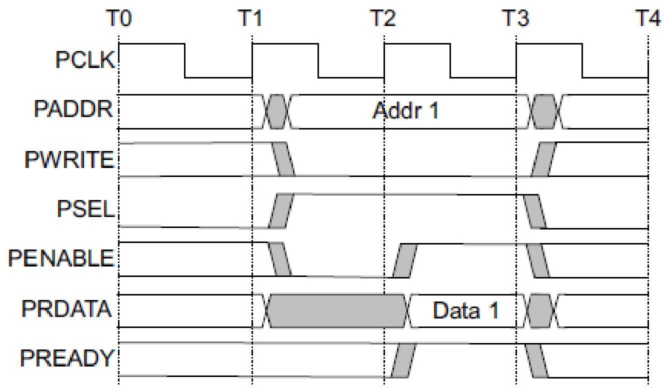

Fig. 4: APB Protocol Signals for Reading

Table 2. Signals in the Configuration Interface

Interfața pentru ușă			
Signal Name	**Number of Bits**	**Direction**	**Description**
led_control	1	Input	We use an LED to indicate whether the oven door is closed or not.

Table 3. The signals within the display interface

The result display interface(READY)			
ready	1	Output	final result

1.4 List of DUT Functionalities

- The DUT must simulate the functionalities of an electric oven, providing the user with operation modes, temperature settings, and a timer. The DUT receives data from the user interface via the APB interface.
- The DUT also checks the temperature. We allocate an 8-bit address for temperatures ranging from 0 to 200 degrees. Four temperature values (from 1 to 4) can be selected, and the temperatures can be changed during the countdown operation. Once the timer reaches its end, the program is considered complete.

1.5 Schematic Diagram

- The timer receives a value between 1 and 255, after which it decrements by 1 unit with each clock cycle until it reaches 0, at which point the timeout becomes 1. Once a value is assigned, a delay is required to allow the time to decrement properly.
- For the operating modes, we allocate values from 1 to 4. When a functionality is selected, that mode is activated. If multiple modules are chosen simultaneously, priority is assigned from the highest value to the lowest. The system will remain in the same operating mode as long as the value is unchanged; if a functionality receives the value 0, the idle state will be activated.

- In the ready module, it checks if an active operating mode exists, the temperature is greater than 0, the timeout equals 1 (indicating the timer has finished), and the door is closed. If all these conditions are met, the system will be set to 1.

Challenges Encountered

- At the beginning of the DUT implementation, instead of testing each module after its completion, We developed all the modules, including the top-level module with the testbench, and only performed a single test at the end with all the combined modules, which, understandably, did not work.
- After completing all the modules, We became stuck for a while on the timer module. It functioned correctly when compiled alone in ModelSim, but failed to operate when compiled with all the modules. After wasting several hours, we concluded that the module was correctly written; the issue lay in the testbench, where we had not allocated sufficient delay for the module.
- I had assumed that my colleague responsible for the APB interface was managing everything related to it, so I did not include it in the DUT. This oversight was brought to my attention later.
- When I began designing the DUT, I did not fully understand the concept of registers and that the design needed to be implemented with the APB interface in mind. As a result, I created a module that was less than ideal, with the implementation of the APB interface being added at the end of the process.

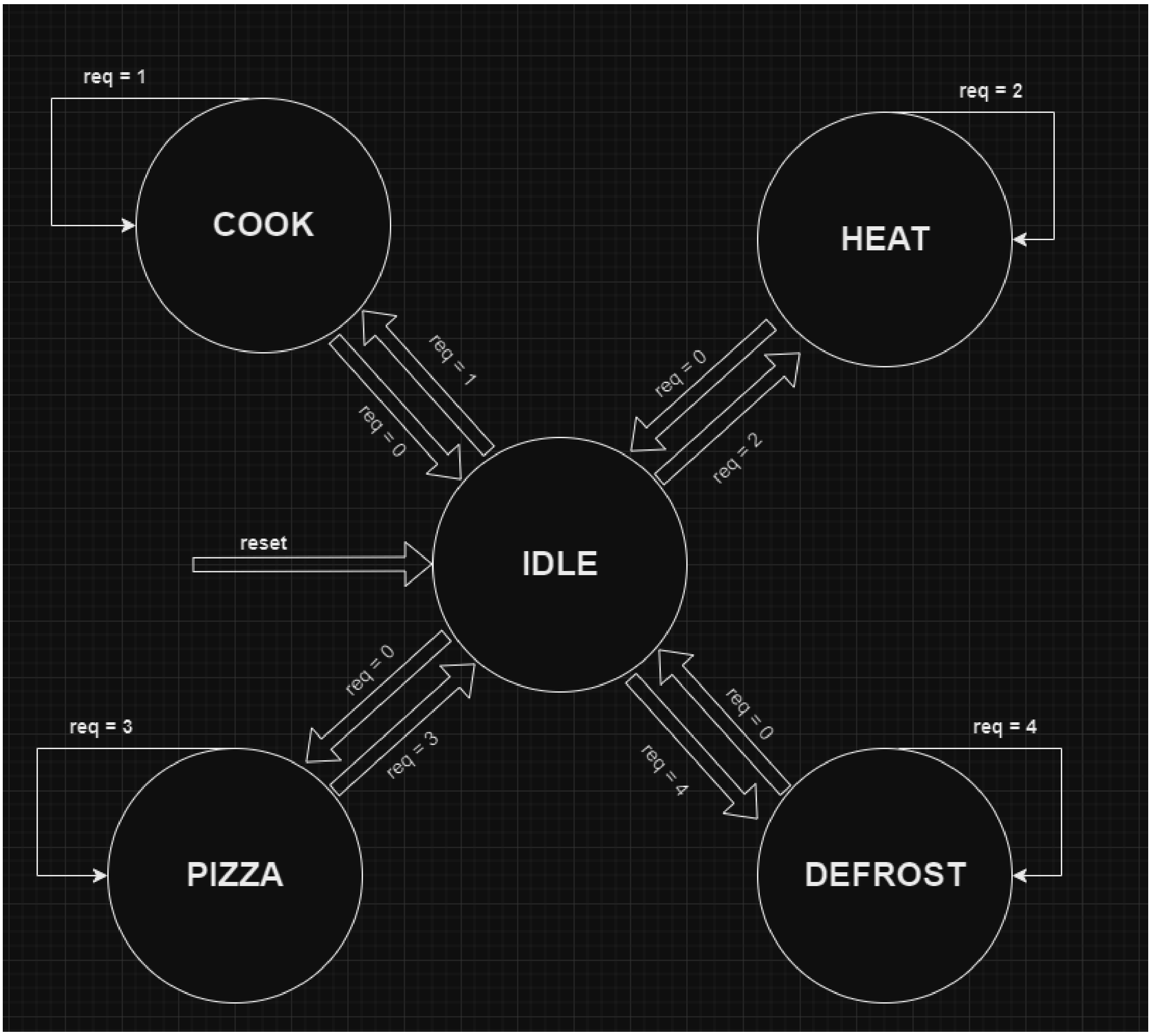

Figure 5. FSM Diagram of the Operating Mode Module

1.6 Waveforms

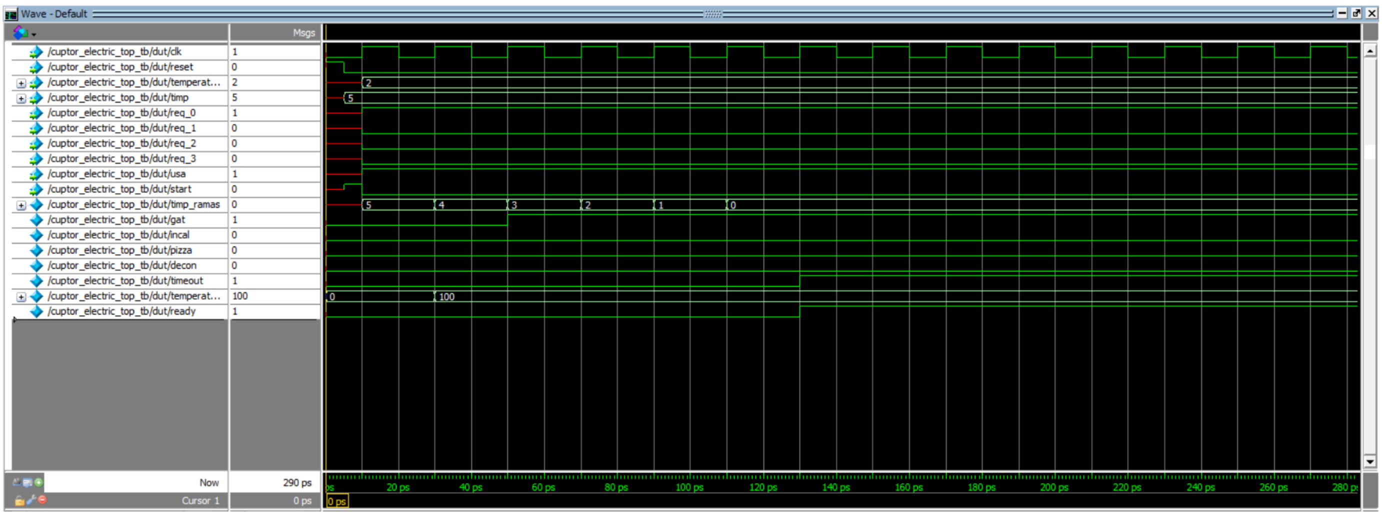

Figure 6. Resulted Waveforms in ModelSim

1.7 Register Table

A table will be created containing the names of the registers, the addresses at which they can be accessed, and their descriptions.

Example:

Table 4. Table of Registers Accessible by the DUT

Register Name	Access Address	Register Structure	Description of Register Fields
set_temperature_reg	0x00		The set temperature can have 5 values ranging from 0 to 4, where the temperature increases by 50 degrees starting from 0.
operation_mode_reg	0x01		This register indicates the mode being used and can take the following values: • IDLE = 3'b000 • COOK = 3'b001 • HEAT = 3'b010 • PIZZA = 3'b011 • DEFROST = 3'b100 A value bigger than 3’b100 in bits [2:0] of the operation_mode_register is ignored Bit 3 represents the start signal from the timer module (used to initiate the countdown timer). This bit automatically toggles to 0 in the next cycle after it was written with 1(First, the register `time_reg` should be written, followed by setting bit 3 to 1.)

time_reg	0x02		This register is used to set the timer value for the oven, which can range from 0 to 255.

2 Verification of the Digital Circuit Project

2.1 Architecture of the Verification Environment

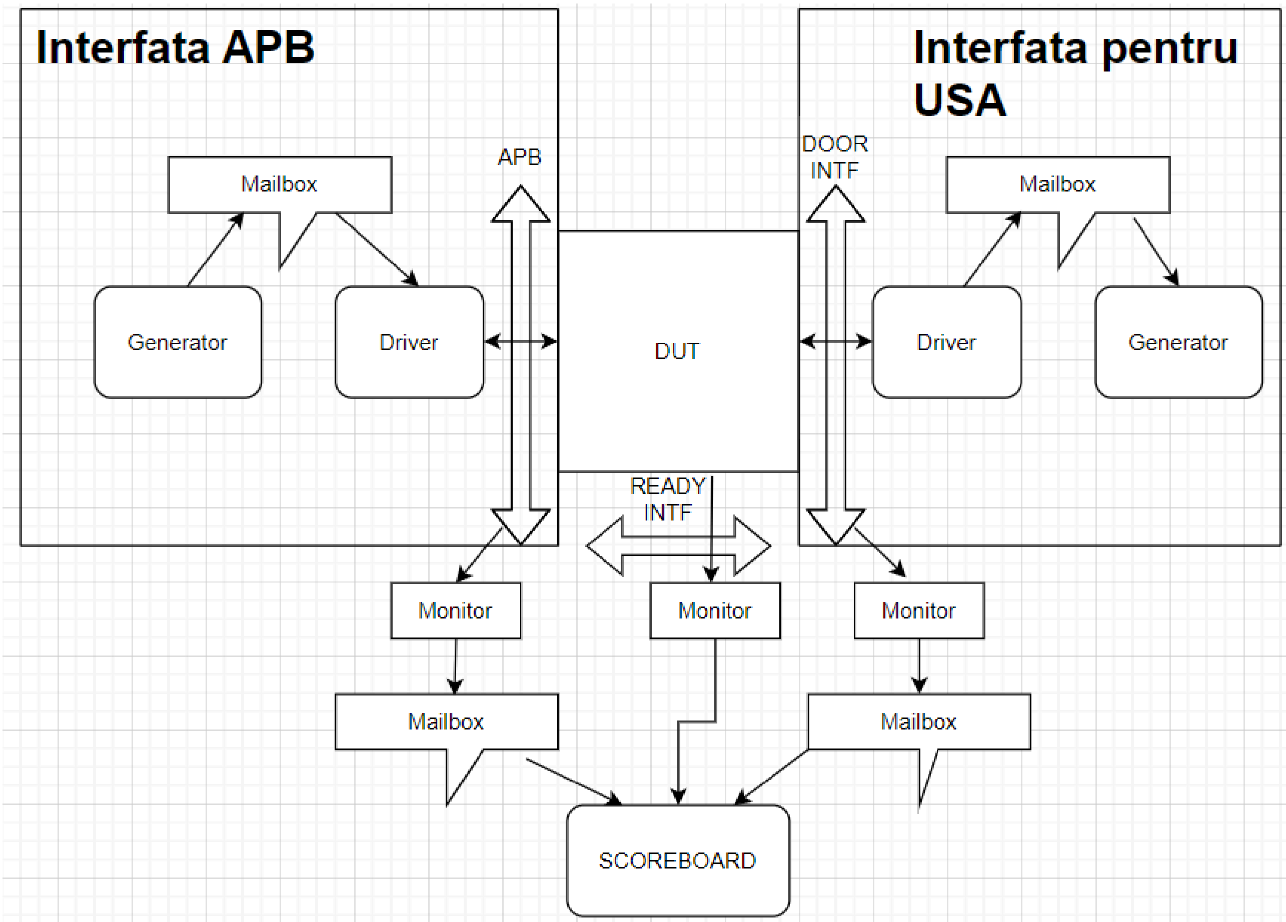

Figure 7. Verification Environment Interface(Modifica la sfarsit daca se adauga/sterg din proiect)

2.1.1 APB Driver

The APB driver retrieves data from the generator at an abstract level and sends it to the DUT according to the communication protocol for that interface. Here, the macro `APB_DRIV_IF` is declared, representing the interface that the driver will use to send data to the DUT. The reset requirements are also defined, initializing each signal. If the driver does not receive data from the generator, it remains in a waiting state until data is received.

2.1.2 Scoreboard / Reference Module(Completeaza aici dupa ce il faci)

2.1.3 Door Driver

The door driver collects transactions from the generator and transmits them to the DUT through an interface. This is accomplished via a virtual interface and a mailbox for receiving data from the generator. The driver operates in two modes: one that waits for the reset signal from the interface and another that sends transactions to the interface. The transmission process is contingent on receiving data from the generator and halts when the reset signal for the interface is activated.

2.1.4 APB Monitor

The APB monitor tracks data traffic on the DUT interfaces, from which verification data is gathered. In this implementation, the data from the interfaces are sent to the scoreboard for verification. The monitor declares and creates a transaction object that will contain the data retrieved from the interface.

2.1.5 Door Monitor

The door monitor tracks data traffic on the DUT's interfaces. It captures verification data and reconstructs the transactions. The data collected from the interfaces is sent to the scoreboard for verification. The monitor connects with the real DUT interface, collecting data and sending it to the scoreboard for verification.

2.1.6 LED Interface Monitor

The led_mintor class monitors data from the DUT interfaces and transmits it to a scoreboard for further processing. It can also be configured to collect coverage data associated with LED monitoring.

2.1.7 Door Monitor

`Door_transaction` represents the transactions between the generator and the driver for the verification system. The attribute `usa_inchisa` stores the door's state and receives random values when using the `randomise` function. The constraint ensures that the generated values are appropriate. In this case, the value 1 represents the door's closed state with a probability of 90%, while the value `0` indicates the open state with a probability of 10%. The `post_randomize` function displays information such as the `usa_inchisa` state to assist with monitoring. This display is helpful for verifying the system's correct operation. The `do_copy` function creates a new object, assigns the attribute values from the current object to the newly created object, and then returns the copied object.

2.1.8 APB Interface Transaction

In the APB interface transaction, the data type used for storing data transferred between the generator and the driver is declared. The monitor retrieves data from the interface and reconstructs it using an object of that data type before processing it. The `apb_transaction` class declares its attributes, and then, using the `rand` keyword, random values are generated through the `randomize()` function. A delay is employed to space the clock cycles between two transactions on the interface. Constraints are enforced by the compiler when random values are generated as a result of using the `randomize` function.

2.1.9 LED Transaction

The `transaction_led` class represents a data structure used in system verification to store and manipulate information exchanged between a generator and a driver. This class manages data associated with an LED and offers functionalities such as generating random values and displaying them during debugging.

2.2 List of Verification Elements

Properties to Verify with Assertions	**Description**
psel_and_penable_active	Verify the validation of the PENABLE signal at the second clock cycle after the PSEL signal is activated.
pready_after_psel_falls	Check if the data transfer has occurred by invalidating the PREADY and PSEL signals.
pwdata_activate_at_psel_and_pwrite	Verify the initiation of the data transfer, specifically if the selection signal PSEL is active (1) and the write data signal is also active, then the write data signal will be activated.
pready_ends	Check if the PREADY signal activates on one clock cycle,

	then at the next clock cycle the signal goes to 0.
prdata_activate	Verify if the PRDATA signal is activated when PSEL is high (1) and PENABLE is low (0).

2.3 Functional Coverage Elements

Event: cover_trans_done_ev: issued at the end of a burst transaction	
Cover Point	**Description**
burst_len_cp	**Burst length** has taken all values from 1 to 16.
burst_addr_cp	The entire address range has been covered. The addresses are 32 bits; the minimum and maximum addresses are considered, and the rest of the range is divided into 32 equal parts.
burst_rd_wr_cp	Both read bursts and write bursts have been made.
cross_burst_len_rd_wr_cp	There is a crossover between **burst_len_cp** and read/write.

2.4 Tests

Table 5. List of implemented tests for verifying the DUT (Device Under Test)

The name of the file that contains the test	**Description of the test**
operation_mod_test.sv	In this test, each mode is changed individually according to Figure 3, and then the selected mode is read.
temperature_test.sv	In this test, the temperature is randomly changed to see if any issues arise during the change.
time_reg_test.sv	In this test, limit values and a few middle values are chosen.
reg_test.sv	In this test, random values are first written to the registers, and then they are written with 0x55 (01010101) and 0xAA (10101010) to

	ensure that each bit in the register can be both 1 and 0.
time_modop_test.sv	During this time, all registers were tested, resulting in the LED (mod_ready) being lit..

2.5 Results Obtained

During the execution of each test, good results were obtained. For the temperature and operating mode module, each value specified in the DUT was tested. For the timer module, limit values and several other values between the limits were tested.