ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ
----------o0o----------

ĐỀ CƯƠNG LUẬN VĂN TỐT NGHIỆP

RENAS MCU

(RISC-V cpu, AMBA bus, SPI …)

GVHD: TS. Trần Hoàng Linh
SVTH: Lê Quang Hưng
MSSV: 1711631

TP. HỒ CHÍ MINH, THÁNG 1 NĂM 2021

LỜI CẢM ƠN

Trong thời gian làm đồ án môn học, em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô, gia đình và bạn bè.

Em xin gửi lời cảm ơn chân thành đến TS. Trần Hoàng Linh, chủ nhiệm bộ môn Điện tử - Trường Đại học Bách Khoa Hồ Chí Minh; anh Nguyễn Trung Hiếu, giảng viên bộ môn Điện tử - Trường Đại học Bách Khoa Hồ Chí Minh đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình thực hiện đồ án môn học.

Em cũng xin chân thành cảm ơn các thầy cô giảng viên trong trường Đại Học Bách Khoa Hồ Chí đã dạy dỗ cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành, anh Nguyễn Hùng Quân, kỹ sư tại Ampere Computing Vietnam đã chia sẽ các kiến thức về kỹ thuật thiết kế phần cứng, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

Tp. Hồ Chí Minh, ngày 13 tháng 1 năm 2021 .

Sinh viên

Lê Quang Hưng

MỤC LỤC

DANH SÁCH HÌNH

DANH SÁCH BẢNG

1. GIỚI THIỆU

1.1 Tổng quan

Semiconductor là một trong những ngành công nghệ trọng điểm của các quốc gia phát triển trên thế giới. Các công ty trong ngành này hầu hết là các tập đoàn lớn với nguồn tài chính dồi dào. Du nhập vào Việt Nam trong khoảng 10 năm trở lại đây, ngành vi mạch đã định hình và phát triển với sự xuất hiện của hàng loạt các tập đoàn vi mạch: Renesas, Ampere, Marvell, Synopsys, ...

Hình 1: Các công ty vi mạch ở Viet Nam (nguồn: Cộng đồng vi mạch Việt Nam)

Quá trình thiết kế và sản xuất vi mạch phải trải qua nhiều giai đoạn từ mức vi kiến trúc hệ thống sau đó là các khâu từ Front-end đến Back-end. Để phục vụ cho mục tiêu nghề nhiệp của em, đề tài luận văn này nhắm vào các khâu quan trọng của Front-end: Logic Design & Design Verification. Các IP được đặt ra thực hiện là các thành phần cơ bản và thường có trong một con chip thực tế.

- CPU
- BUS
- PERIPHERAL

1.2 Tình hình nghiên cứu trong và ngoài nước

Nguồn nhân lực vi mạch có kinh nghiệm ở Việt Nam tuy có thể đáp ứng được yêu cầu công việc, nhưng số lượng này lại không nhiều. So sánh Job Description ở vị trí Fresher của một ứng viên mới tốt nghiệp đại học ở Việt Nam với các nước lân cận như: Singapore, Malaysia, ... ta có thể thấy được một sự chênh lệch trình độ rất lớn ở đây.

Một trong những lý do khiến cho cộng đồng vi mạch ở Việt Nam không phát triển bằng các nước khác là do nguồn tại liệu tiếng Việt cho ngành này hiện nay vẫn còn quá ít. Các dự án mã nguồn mở hoặc nghiên cứu, không nhiều và không đủ tiếng vang để thu hút các bạn trẻ tham gia.

Những năm gần đây. với sự trưởng thành và nhận thức của các anh chị kỹ sư dày dặn kinh nghiệm. Các chương trình xây dựng nguồn nhân lực và chia sẻ kiến thức đã được đề ra và thực hiện: khóa học cấp chứng chỉ của Synopsys (anh Nguyễn Thanh Yên - Synopsys - Hà Nội), dự án Vanguard SoC (anh Nguyễn Hùng Quân - Ampere - Hồ Chí Minh), ...

Các môn học ở bộ môn Điện-Điện tử trường đai học Bách Khoa Hồ Chí Minh đã được thiết kế và điều chỉnh để có thể phù hợp với nhu cầu thị trường và giúp sinh viên tiếp cận với các kiến thức nền tảng.

1.3 Mục tiêu đề tài

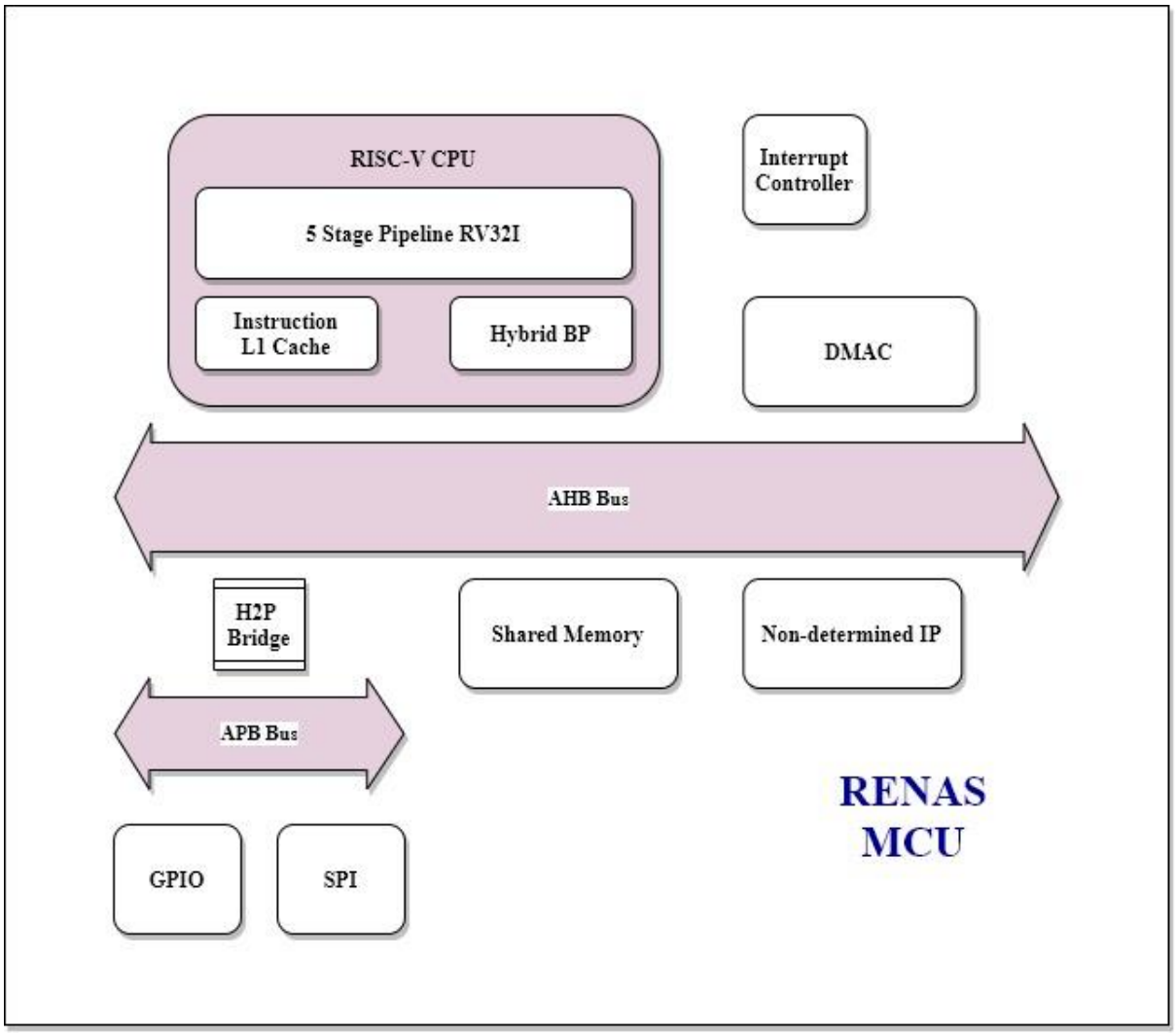
Với các vấn đề được nêu ở trên, đề tài luận văn của em được thiết kế dựa trên các yêu cầu công việc của một kỹ sư Front-end. Các mục tiêu được đặt ra đó chính là:

• Logic Design: hoàn thành RTL code và specification cho các khối chức năng.

• Design Verification: xây dựng một số môi trường mô phỏng cho các khối chức năng.

• Verification: kiểm tra tính đúng đăn của các khối chức năng. Một thiết kế kiến trúc hệ thống, hay một mô tả RTL dù có tốt đến đâu, đều không thể được sử dụng nếu nó không thể chứng minh tính đúng đắn của mình.

2. NỘI DUNG ĐỀ TÀI



Hình 2: Renas MCU

Renas MCU là một microcontroller unit sử dụng lõi cpu RISC-V và các ngoại vi được kết nối bởi hệ thống AMBA bus (AHB và APB).

• Nội dung 1: AMBA Bus & Constraint Random Verification

+ Tìm hiểu nguyên lý, lý thuyết về Multilayer-AHB bus và APB bus.

+ Xây dựng sơ đồ khối và RTL code.

+ Tìm hiểu lý thuyết về SystemVerilog for Verification và môi trường Constraint Random Verification.

+ Xây dựng môi trường mô phỏng Constraint Random Verification

+ Kiểm định hệ thống bus bằng môi trường mô phỏng CRV trên với mục tiêu: đảm bảo functional hệ thống, coverage ít nhất là 80%.

• Nội dung 2: SPI & UVM

+ Modify RTL code của SPI peripheral đã được thực hiện ở các kì trước.

+ Tìm hiểu lý thuyết về UVM

+ Xây dựng môi trường mô phỏng UVM cho SPI.

+ Kiểm định SPI bằng môi trường mô phỏng trên với mục tiêu: đảm bảo functional hệ thống, coverage ít nhất là 80%.

• Nội dung 3: RISC-V cpu

+ Modify cấu trúc RISC-V cpu ở đồ án học kì 192 (RVS192) theo hướng tối ưu và phù hợp với cấu trúc hệ thống Renas MCU.

+ Xây dựng môi trường mô phỏng CPU.

+ Kiểm định cpu RISC-V bằng các bài test chuẩn hóa của RISC-V International.

• Nội dung 4: Shared Memory & Renas MCU version 0.1

+ Xây dựng Shared Memory quản lý bộ nhớ instruction và data của Renas MCU có thể giao tiếp với chuẩn AHB.

+ Kết nối các khối đã được xây dựng ở trên thành Renas MCU version 0.1

+ Kiểm tra functional của Renas MCU version 0.1

• Nội dung 5: Interrupt Controller & Renas MCU version 0.2

+ Xây dựng RTL code và specification cho Interrupt Controller.

+ Kết nối Interrupt Controller ở trên để hoàn thành Renas MCU version 0.2

+ Modify AMBA Bus để tăng hiệu xuất hệ thống.

+ Xây dựng test plan để kiểm định hệ thống, coverage ít nhất 80%

3. GIẢI PHÁP THỰC HIỆN

3.1 Nội Dung 1: AMBA Bus & Constraint Random Verification

3.1.1 AMBA Bus

- Tài liệu tham khảo của AMBA Bus được lấy từ trang chủ của ARM [1].
- Để tối ưu hóa hiệu suất truyền nhận dữ liệu phù hợp với kiến trúc của Renas MCU, chuẩn AMBA Multi-layer AHB cùng với AMBA4 APB được sử dụng.
- **AMBA Multi-layer AHB:**
    - Kiểu interconnect này cho phép các đường truy cập dữ liệu diễn ra song song với nhiều Master và Slave khác nhau tương thích với chuẩn AHB-lite.

- Các bộ phân xử quyền truy cập sẽ được đặt ở phía Slave với 3 cấu hình được sử dụng: Round Robin Arbitrator, Fixed Priority Arbitrator và Dynamic Priority Arbitrator.
- Chuẩn AHB được sử dụng để kết nối giữa cpu với các ngoại vi và bộ nhớ của Renas MCU.

- **AMBA APB:**
  - Đây là chuẩn Bus được sử dụng để quản lý các ngoại vi và thanh ghi cấu hình của Renas MCU. APB Bus được thiết kế không cho phép truy cập các ngoại vi song song cùng lúc. Tại mỗi thời điểm Master của nó chỉ được phép đọc ghi SPI, GPIO hoặc các thanh ghi cấu hình của Renas MCU.

3.1.2 Constraint Random Verification

- Tài liệu tham khảo để xây dựng môi trường CRV được lấy từ sách: ”SystemVerilog for Verification: A Guide to Learning the Testbench Language Features - Chris Spear” [2].
- Môi trường CRV cho phép random các cấu hình và dữ liệu truyền nhận trên các chuẩn giao tiếp Bus.
- Thay vì chỉ sử dụng eye-check để kiểm tra waveform, phương pháp CRV sẽ hiện thực một môi trường mô phỏng với nhiều tầng trừu tượng khác nhau.
- Các tầng cấu trúc của CRV dùng để mô phỏng AHB và APB bus là giống nhau. Tuy nhiên ứng với từng chuẩn giao tiếp, các Driver, Monitor và các giá trị Constraint Random phải tương ứng với từng chuẩn Bus. Môi trường CRV sử dụng các điểm mạnh của SystemVerilog giúp tái sử dụng môi trường mô phỏng AHP và APB với ít thay đổi nhất có thể.

3.2 Nội dung 2: SPI & UVM

3.2.1 SPI

- Design Specification và RTL code cho RTL được lấy từ một dự án riêng đã được thực hiện từ trước, tuy nhiên vẫn chưa được verify hoàn chỉnh.
- Thiết kế này hỗ trợ giao tiếp với chuẩn APB, cho phép cấu hình các thanh ghi chức năng SPI.

3.2.2 UVM

- Môi trường UVM verify SPI được tham khảo từ môi trường UVM verify UART:

https://nguyenquanicd.blogspot.com/2019/05/uvm-bai-1-mo-ta-tong-quan-va-mo-ta-loi.html

3.3 Nội dung 3: RISC-V cpu (Renas cpu)

- Modify cấu trúc RISC-V cpu ở đồ án học kì 192 (RVS192) theo hướng phù hợp với cấu trúc hệ thống Renas MCU:
  - Các lệnh yêu cầu data hay instruction nếu cần truy cập bộ nhớ sẽ gửi yêu cầu đọc xuống AHB bus. Tùy theo loại hay tính chất dữ liệu cần đọc mà dữ liệu request tương ứng sẽ được tạo ra:

    Vd: yêu cầu đọc data sẽ được chuyển thành AHB request với hburst = SINGLE (do RISC-V cpu sử dụng ở RENAS 192 không hỗ trợ Data Cache).
- Kiểm tra cpu RISC-V bằng các bài test của RISC-V international.
- Xây dựng môi trường mô phỏng CPU có khả năng thu thập các thông tin về hiệu suất hay CPI của Renas cpu.

3.4 Nội dung 4: Shared Memory & Renas MCU version 0.1

- Level 2 Cache của RVS192 sẽ được chuyển thành Shared Memory với một số thay đổi để có thể giao tiếp với chuẩn AHB.
- Kết nối các khối đã được xây dựng ở trên thành Renas MCU version 0.1.
- Kiểm tra functional của Renas MCU version 0.1
  - Cài đặt các thanh ghi cấu hình.
  - Giao tiếp SPI.
  - Giao tiếp Shared Memory.
  - Giao tiếp GPIO.

3.5 Nội dung 5: Interrupt Controller & Renas MCU version 0.2

- RTL code và Specification cho Interrupt Controller (ICC) được xây dựng dựa trên IP ECM (Error Control Management) được thực hiện ở kì thực tập 193.

- Interrupt Controller cho phép chọn các điều kiện cho phép dừng CPU hay reset lại hệ thống:

  Vd: khi phát hiện ra một lệnh bị dừng quá lâu ICC sẽ reset hệ thống.

- Modify RISC-V CPU và AHB Bus để tăng hiệu suất hệ thống: sử dụng các hỗ trợ cacheable, bufferable, … của AHP5.

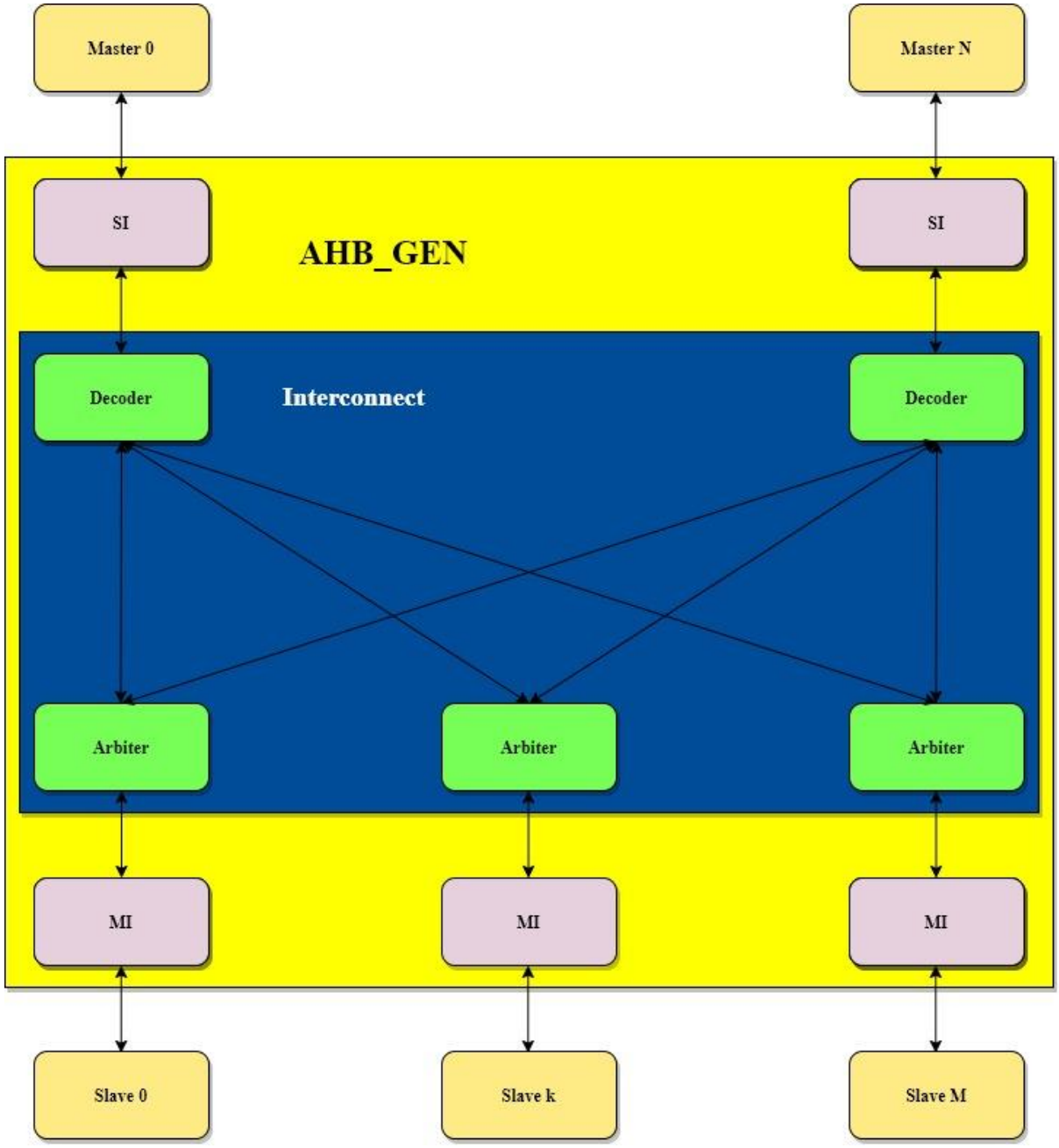
4. DỰ KIẾN KẾT QUẢ ĐẠT ĐƯỢC

4.1 Kết quả sơ khởi đã đạt được

4.1.1 Nội Dung 1: AMBA Bus & Constraint Random Verification

AHB_GEN_201 version 0.1

- Để hỗ trợ cho các thay đổi sau này nếu cần thiết, AHB_GEN_201 được sử dụng như một tool hỗ trợ giúp tạo RTL code từ các cài đặt được định nghĩa.
- Các thống số cần cài đặt là
  - Số lượng Master và Slave.
  - Tầm địa chỉ của từng Slave.
  - Các Master và Slave nào được phép giao tiếp với nhau.
  - Chế độ phân xử được sử dụng trước mỗi Slave.
- RTL của AHB Interconnect được tạo ra sẽ có cấu trúc tương tự như hình minh họa bên dưới:



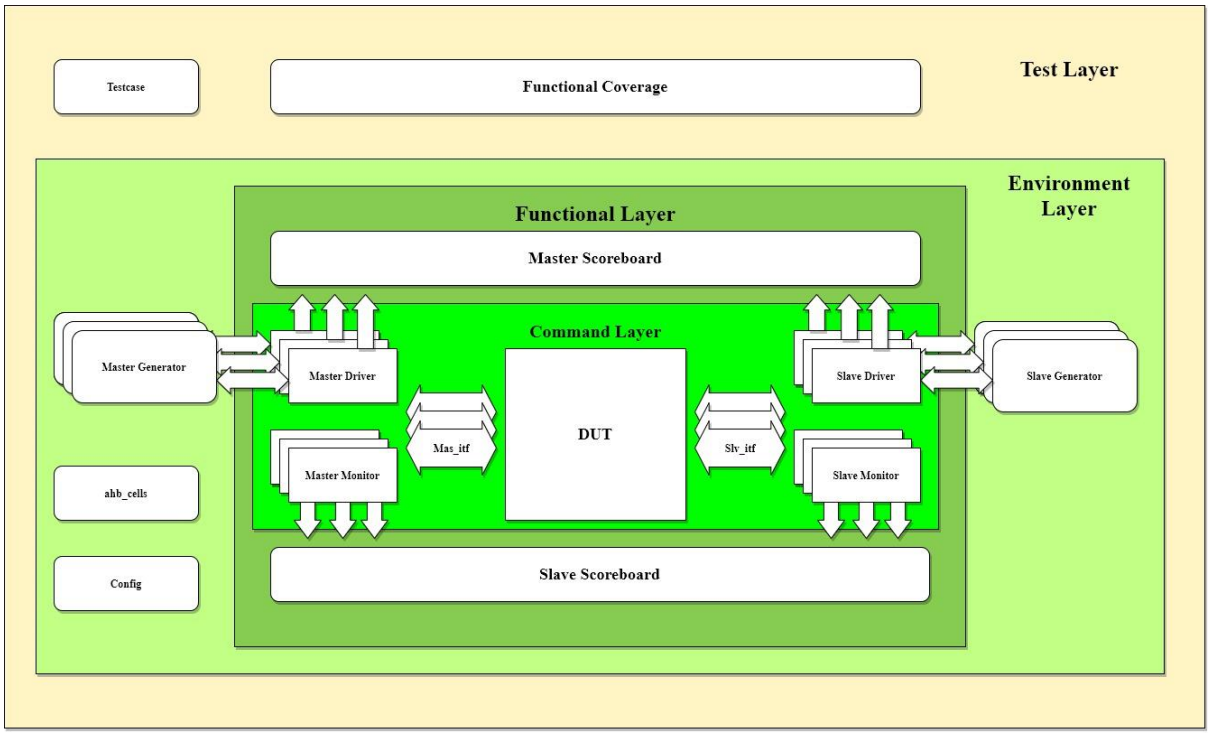
Hình 3: AHB_GEN block diagram

CRV Environment version 0.0 (AMBA AHB)

- Môi trường mô phỏng CRV này được xây dựng nhằm mô phỏng RTL code tạo ra từ AHB_GEN_201 tool. Do số lượng Master, Slave hay tầm địa chỉ của AHB bus được tạo ra có thể thay đổi tùy theo cài đặt của AHB_GEN_201. Việc viết test case mô phỏng cho từng trường hợp của từng cấu hình được tạo ra của AHB bus là không hiệu quả. Môi trường CRV đươc tạo ra để giải quyết vấn đề trên. CRV cho phép:
    - Số lượng Master và Slave được sử dụng là không giới hạn.

- Kiểm tra các tổ hợp dữ liệu truyền trên AHB bus mà không cần phải viết từng package dữ liệu, và đặt biệt không phải quan tâm đến timing cho mỗi test pattern mới.
- Mô phỏng các Master và Slave giả để giao tiếp với AHB bus.
- Khi người dùng muốn kiểm tra một corner test pattern, thay vì phải viết lại toàn bộ pattern, chúng ta chỉ cần thêm các constraint ràng buộc cho môi trường.

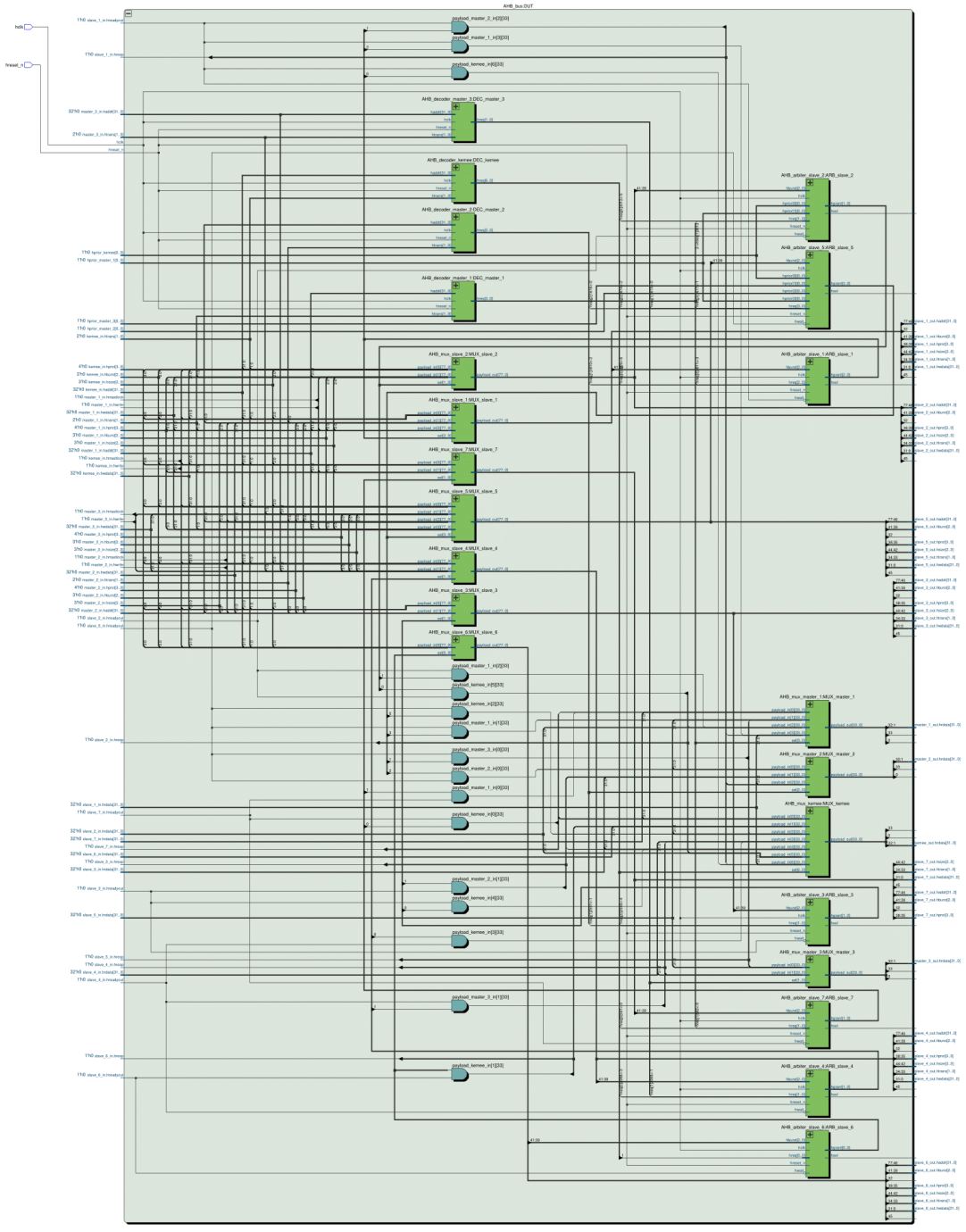
- Các trường hợp lỗi có thể phát hiện bởi CRV:
  - **Cells remaining in Master[x] scoreboard at the end of the test**: dữ liệu được gửi đi ở Master nhưng không nhận được ở Slave.
  - **Cells not found because the scoreboard for Master/Slave[x] empty**: dữ liệu nhận được ở Slave/Master nhưng Master/Slave định danh không gửi dữ liệu này.
  - **Master/Slave cells miss**: dữ liệu nhận được ở Slave/Master nhưng Master/Slave định danh không gửi dữ liệu này.



Hình 4: CRV Environment

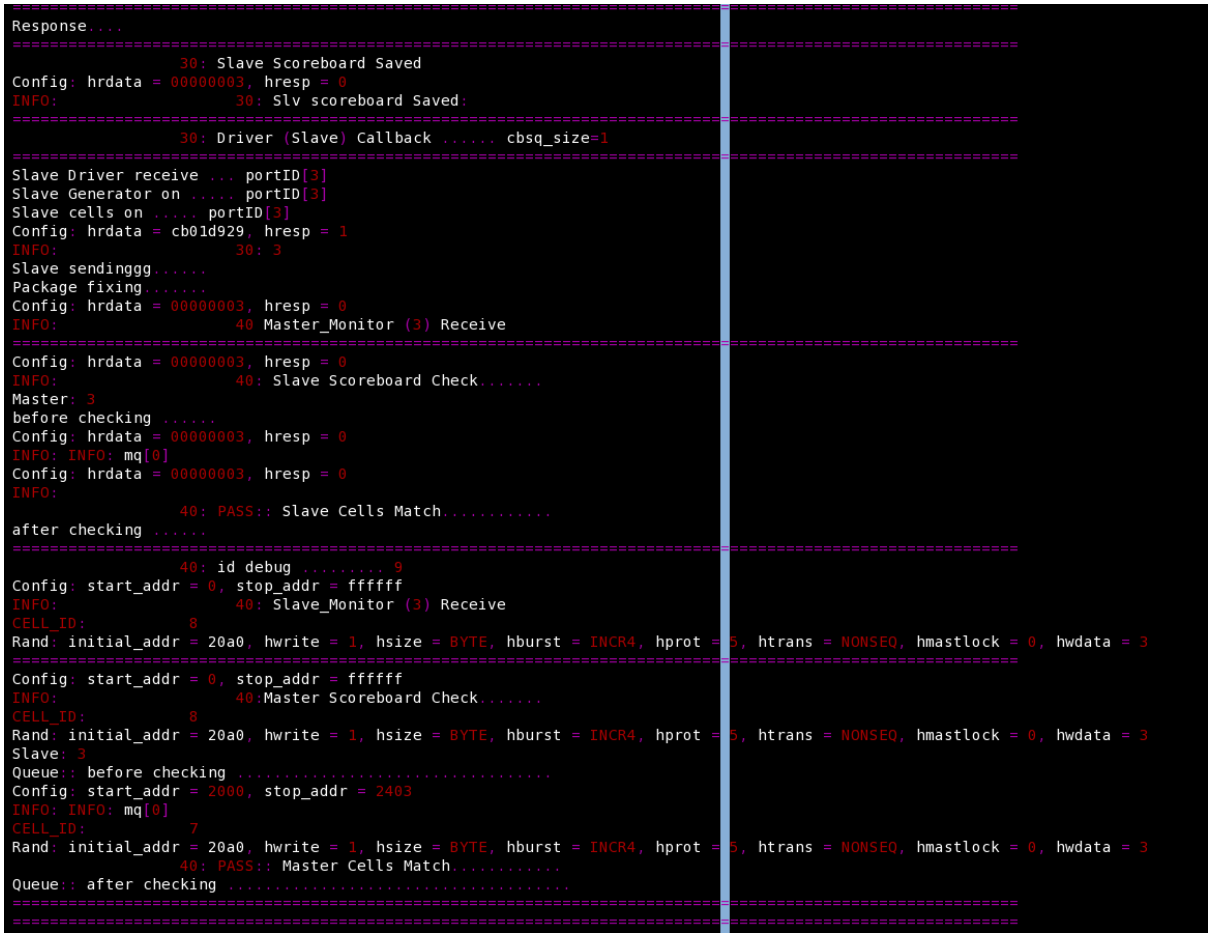
Kết quả mô phỏng (AMBA AHB)

- Ở thời điểm thực hiện báo cáo đề cương này, các test items cho AHB bus vẫn chưa được thực hiện hết.
- Dưới đây là minh họa cho một AHB bus được tạo ra và kết quả mô phỏng của nó bằng CRV (4 Master – 7 Slave).

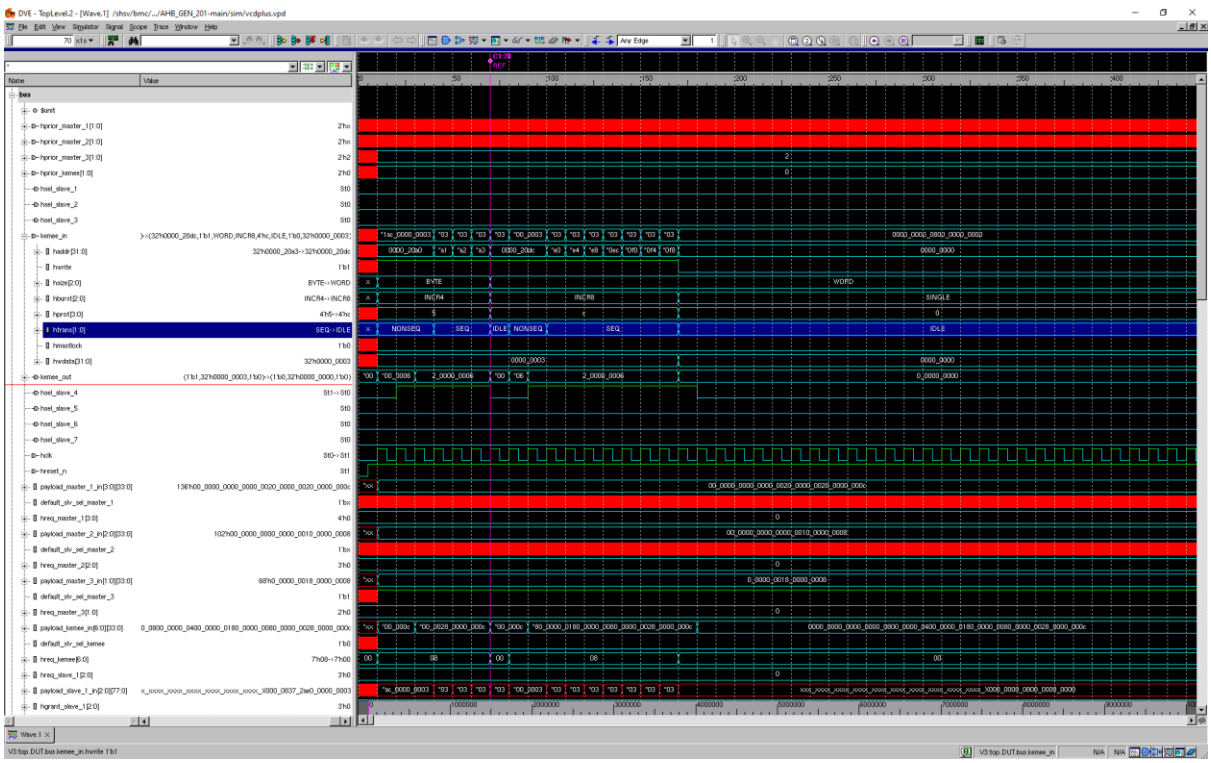


Hình 5: AHB Bus RTL viewer (Quartus)

- Môi trường CRV được chạy trên VCS (11/01/2021)
    - Bug được phát hiện bởi môi trường CRV: sử dụng log file t có thể check ra lỗi ERROR tại các thời điểm tương ứng, dựa vào đó ta có thể mở waveforms để xác định nguyên nhân gây ra bug.



Hình 6: Log file để debug AHB Bus (VCS – 11/01/2021)



Hình 7: Waveform debug AHB bus (VCS - 11/01/2021)

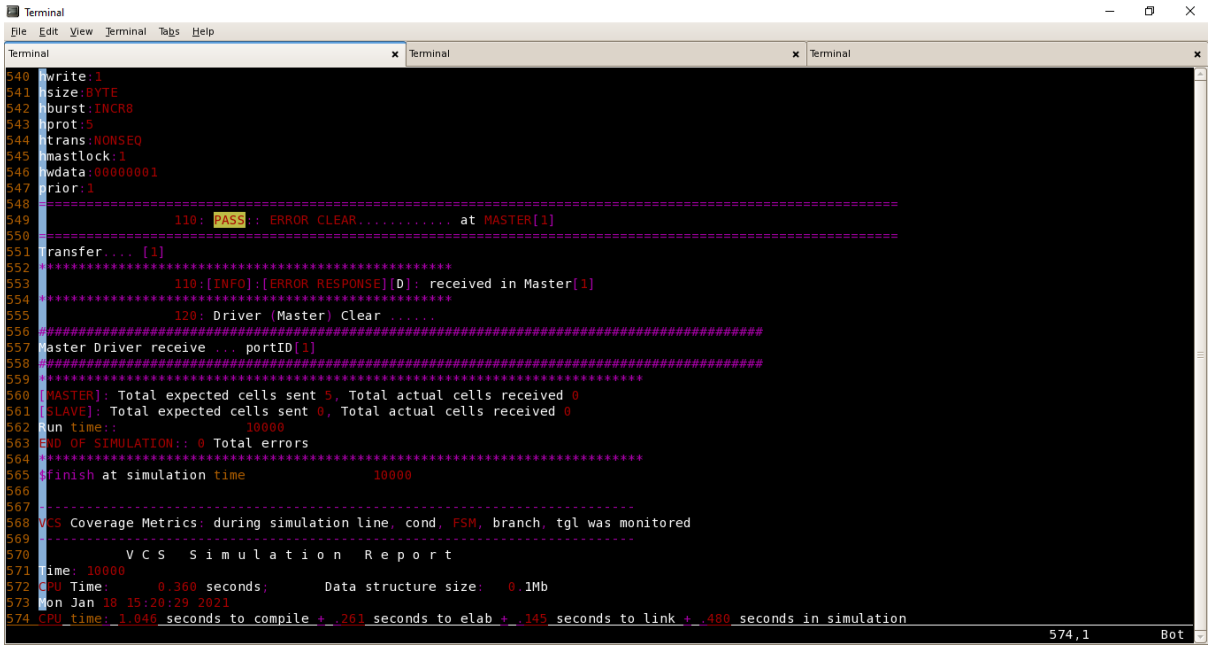
```
**********************************************************************
[MASTER]: Total expected cells sent 13, Total actual cells received 12
[MASTER]:[ERROR]: cells remaining in Master[2] scoreboard at the end of the test
[SLAVE]: Total expected cells sent 12, Total actual cells received 12
Run time::              10000000
END OF SIMULATION:: 1 Total error
**********************************************************************
$finish at simulation time              10000000

------------------------------------------------------------------------
VCS Coverage Metrics: during simulation line, cond, FSM, branch, tgl was monitored
------------------------------------------------------------------------
          V C S   S i m u l a t i o n   R e p o r t
Time: 10000000
CPU Time:     10.090 seconds;       Data structure size:   0.1Mb
Tue Jan 12 11:53:46 2021
CPU time: 1.741 seconds to compile + .414 seconds to elab + .227 seconds to link + 10.288 seconds in simulation
```

Hình 8: Log file debug AHB bus (VCS - 11/01/2021)

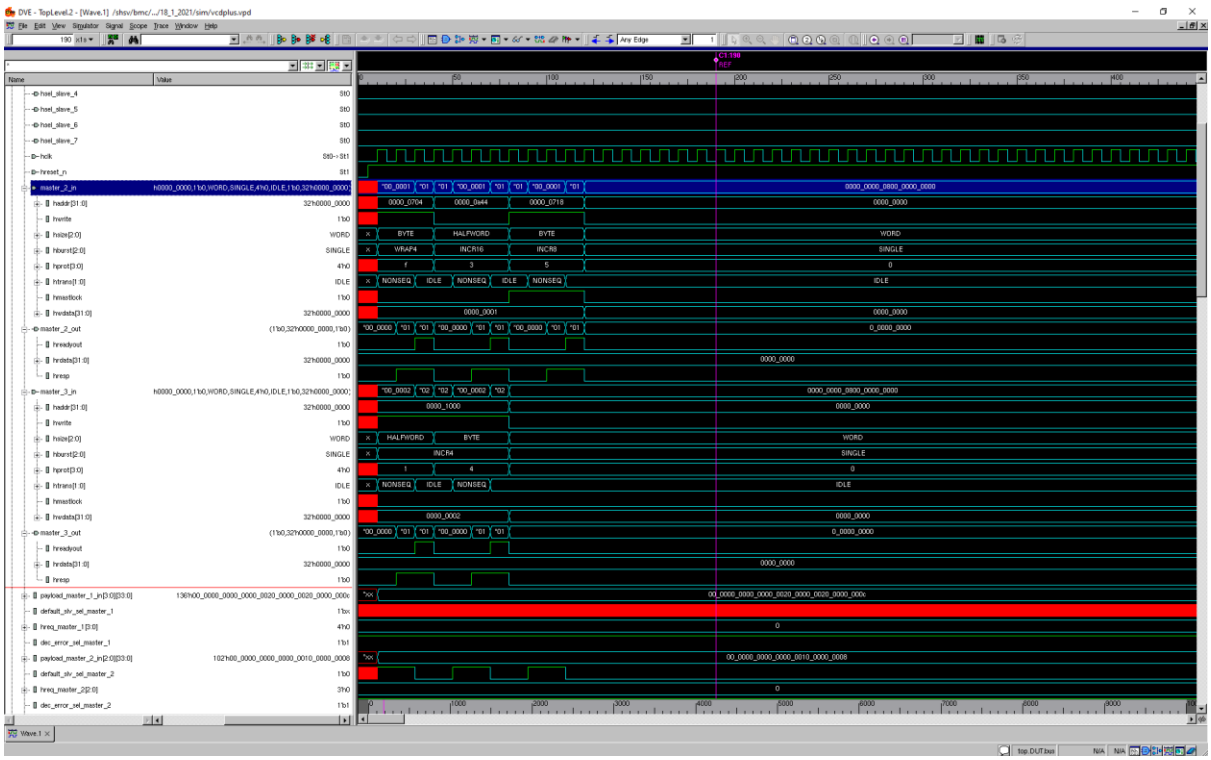
- Môi trường CRV được chạy trên VCS (18/01/2021)

```
########################################################################
[SYSTEM SEED]Simulation run with random seed = 13
########################################################################
========================================================================
[ENV]:[INITIAL CONFIG]: 4 masters, 7 slaves

master in used = 1, cells = 3, prio = 1
master in used = 2, cells = 2, prio = 0


========================================================================
                    0: Build.............
```

Hình 9: Config môi trường CRV trên VCS (18/01/2021)



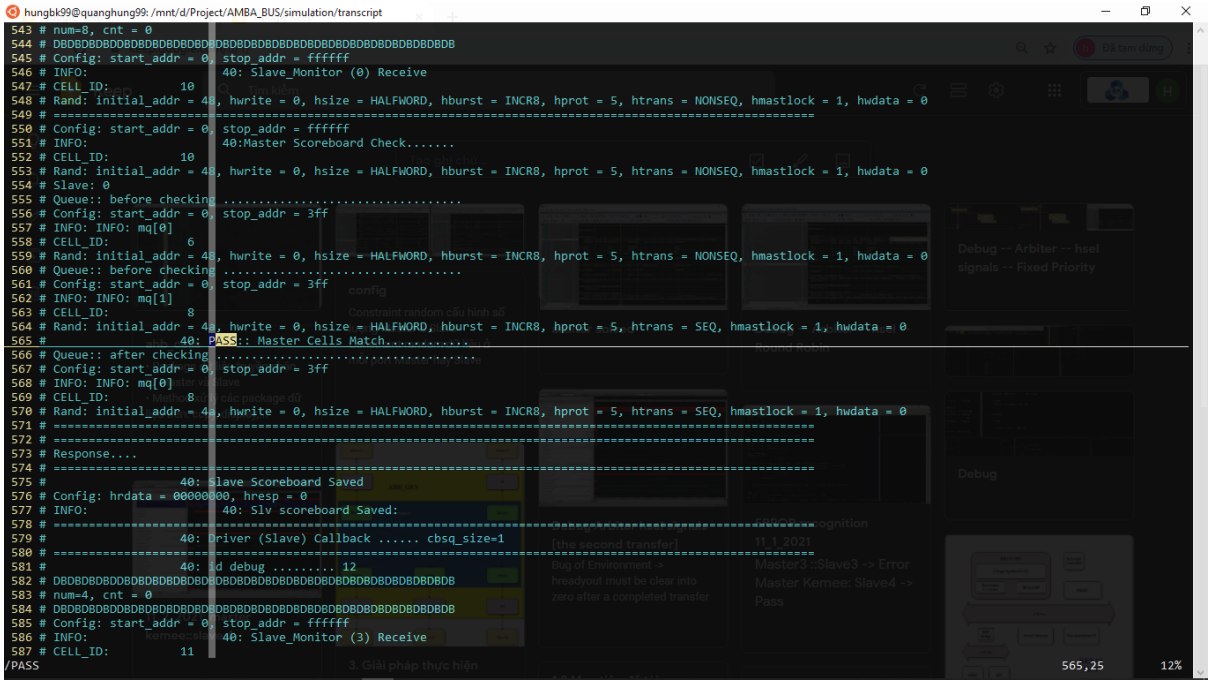
Hình 10: Log file debug AHB bus (VCS - 18/01/2021)



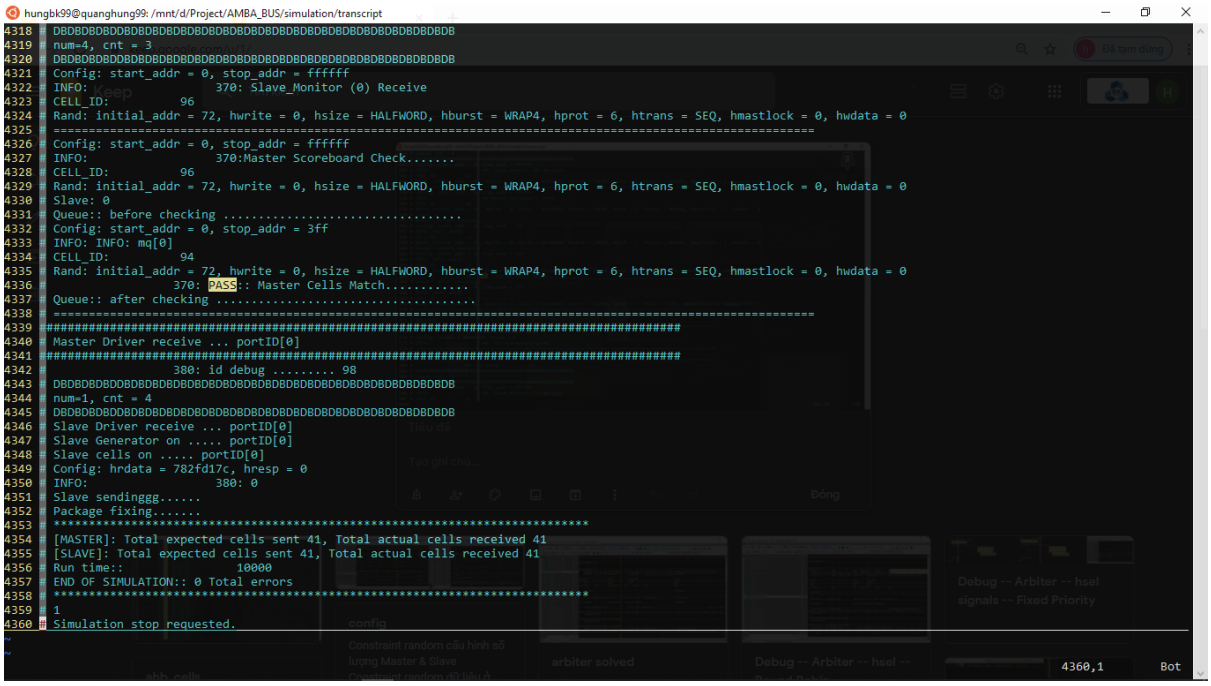
Hình 11: Waveform debug AHB bus (VCS - 18/01/2021)

⇨ *Các trường hợp mô phỏng được nêu ở lần chạy này giúp kiểm tra hoạt động của AHB Bus khi địa chỉ haddr đưa vào Master không thuộc vùng địa chỉ của bất kì Slave giao tiếp tương ứng nào.*

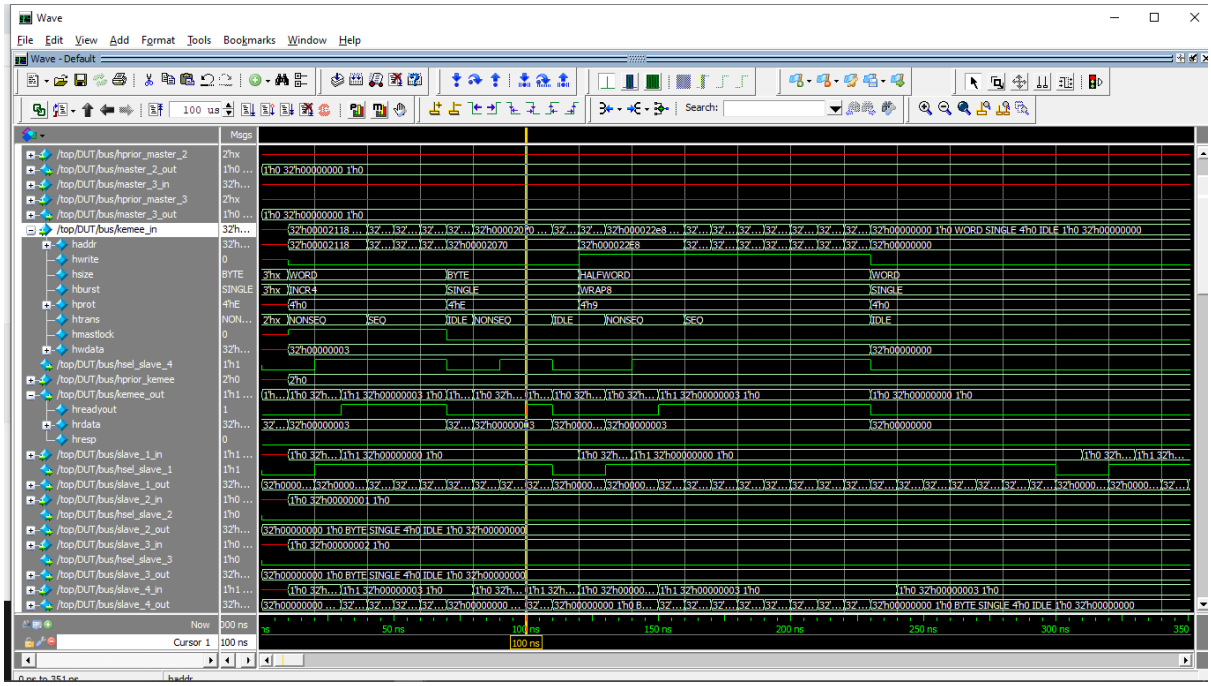
- Môi trường CRV được chạy trên QuestaSim (12/01/2021)



Hình 12: Log file debug AHB bus (QuestaSim – 12/01/2021)

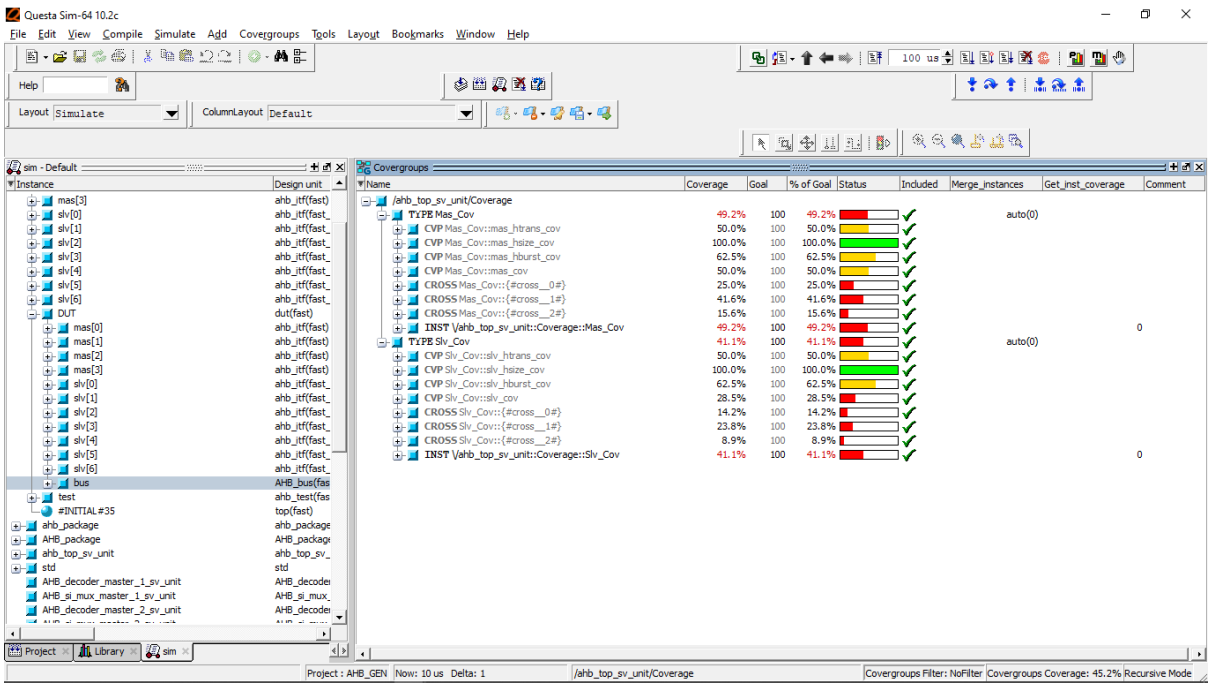


Hình 13: Log file - Result – 1 seed (QuestaSim 12/01/2021)



Hình 14: Waveform debug AHB bus – 1 seed (QuestaSim 12/01/2021)

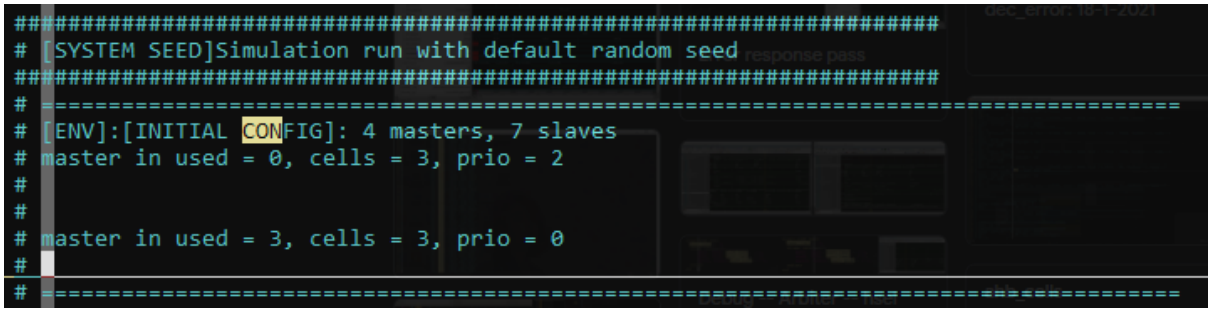
- Môi trường CRV được chạy trên QuestaSim (18/01/2021)



Hình 15: Coverage report trên QuestaSim (18/01/2021)

⇨ Các giá trị Coverage cần quan tâm ở đây là các Cross Coverage.

⇨ Ta có thể thấy coverage hiện tại vẫn còn thấp. Nguyên nhân là do thông tin coverage trên mới chỉ được đánh giá dựa trên một seed chạy đầu tiên.

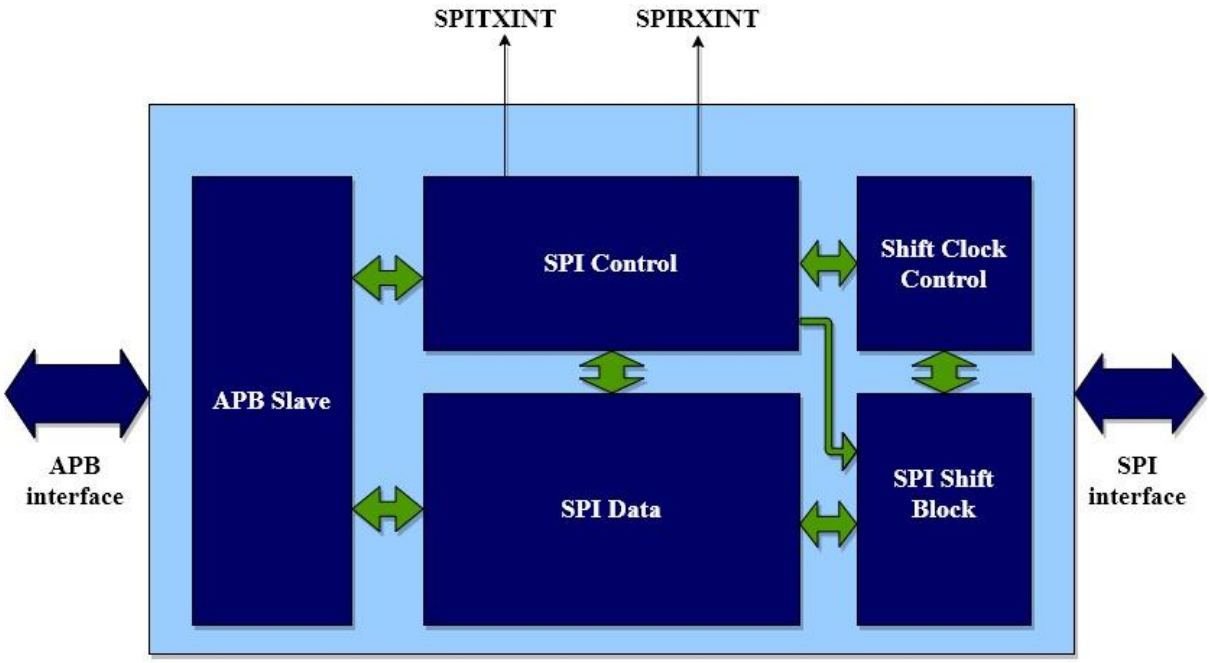


Hình 16: Config môi trường CRV trên QuestaSim (18/01/2021)

4.1.2 Nội dung 2: SPI & UVM

SPI là một chuẩn truyền nhận nối tiếp đồng bộ cho giao tiếp giữa vi điều khiển và các ngoai vi. Một số đặc điểm được thiết kế SPI này hỗ trợ:

- Chế độ Master và Slave
- Chế độ truyền nhận bằng FIFO
- Cấu hình độ rộng của chuỗi dữ liệu truyền nhận (0:32 bit)
- Cấu hình độ trễ giữa các lần nhận và truyền dữ liệu mới khi truyền nhận các gói dữ liệu liên tục
- Cấu hình 256 tốc độ truyền nhận khác nhau
- Cấu hình MSB hay LSB truyền trước
- Hỗ trợ giao tiếp APB4
- Hỗ trợ giao tiếp với nhiều Slave (4)
- 4 định dạng clock truyền nhận (cài đặt thông qua 2 bit CPHA và CPOL)

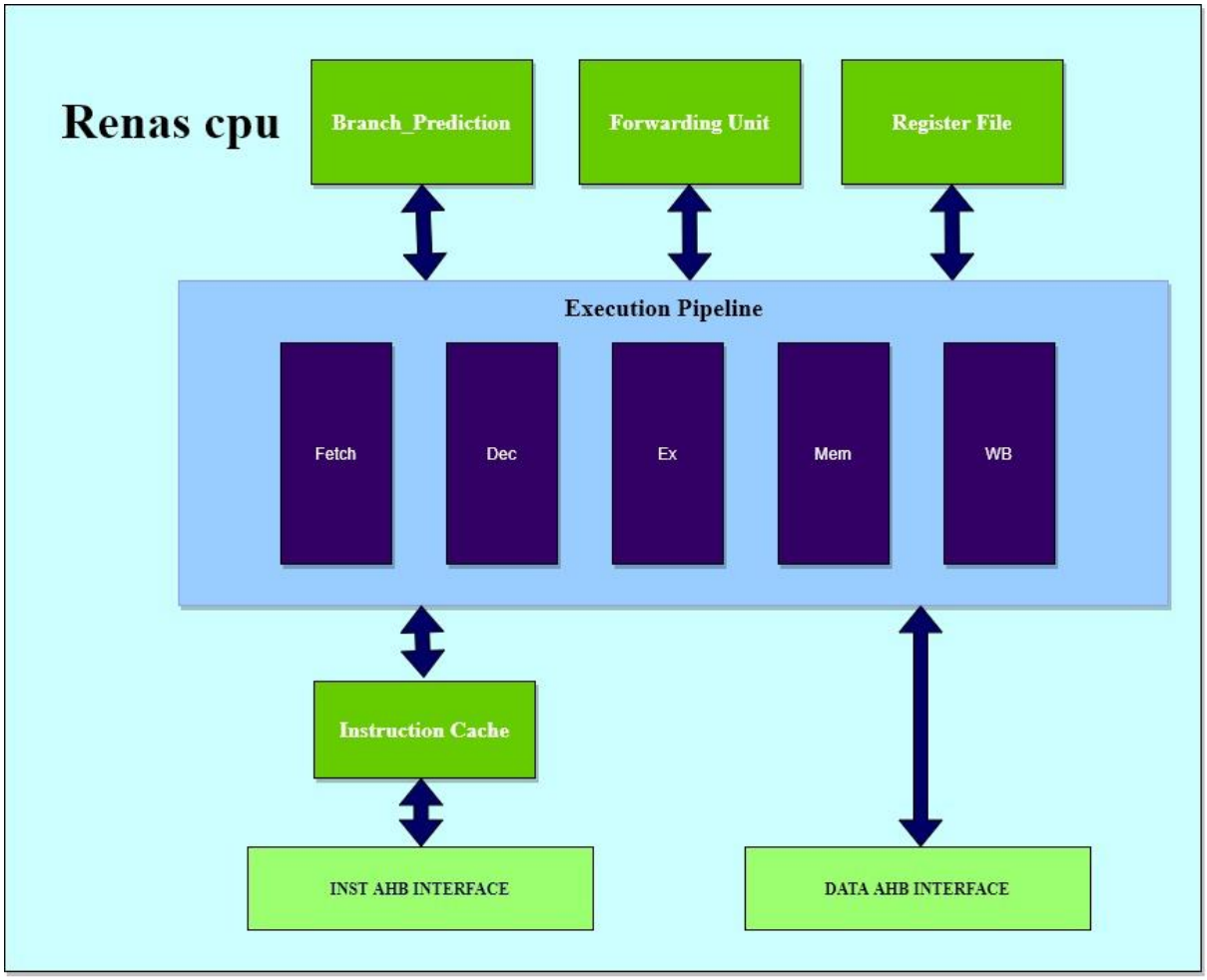


Hình 17: SPI block diagram

- RTL code cho SPI hiện tại đã hoàn thiện, tuy nhiên vẫn chưa được verify.
- Em đang trong quá trình nghiên cứu UVM, do đó vẫn chưa có môi trường nào được thực hiện.

4.1.3 Nội dung 3: RISC-V cpu (Renas cpu)

- Renas cpu ở những version đầu tiên sẽ sử dụng lại RVS192 cpu được thực hiện ở đồ án trước, tuy nhiên L1 Data Cache và L2 Cache sẽ được lượt bỏ.
- Các bộ Interface hỗ trợ giao tiếp với chuẩn AHB sẽ được thêm vào.
- Nội dung này vẫn chưa được thực hiện.



Hình 18: Renas cpu structure dự kiến

4.1.4 Nội dung 4: Shared Memory & Renas MCU version 0.1

- Nội dung này vẫn chưa được thực hiện.

4.1.5 Nội dung 5: Interrupt Controller & Renas MCU version 0.2

- Nội dung này vẫn chưa được thực hiện

4.2 Kết quả dự kiến đạt được

Đối với những nội dung chưa hoàn thành, kết quả dự kiến sẽ đạt được khi tiến hành thực hiện luận văn sẽ giống như mục tiêu đề tài được đặt ra.

5. KẾ HOẠCH THỰC HIỆN

Kế hoạch:

Nội dung	Tháng 2				Tháng 3				Tháng 4				Tháng 5			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	→															
2	→				→											
3					→				→							
4					→				→							
5													→			

Bảng 1: Kế hoạch thực hiện luận văn

6. TÀI LIỆU THAM KHẢO

[1] ARM Ltd, https://developer.arm.com/architectures/system-architectures/amba/specifications

[2] Chris Spear, “SystemVerilog for Verification: A Guide to Learning the Testbench Language Features”.