Hướng dẫn test core Single Cycle

Vy Luong

> **Note**
>
> **Mục đích của tài liệu hướng dẫn này là giúp sinh viên hiểu cách chạy các bài test và sử dụng testbench mẫu do nhóm cung cấp.**
> **Các bạn hoàn toàn có thể tự viết và thiết lập testbench riêng miễn là CPU RTL của bạn có thể chạy và vượt qua các bài kiểm thử được quy định trong tài liệu này.**

1. Cấu trúc testbench

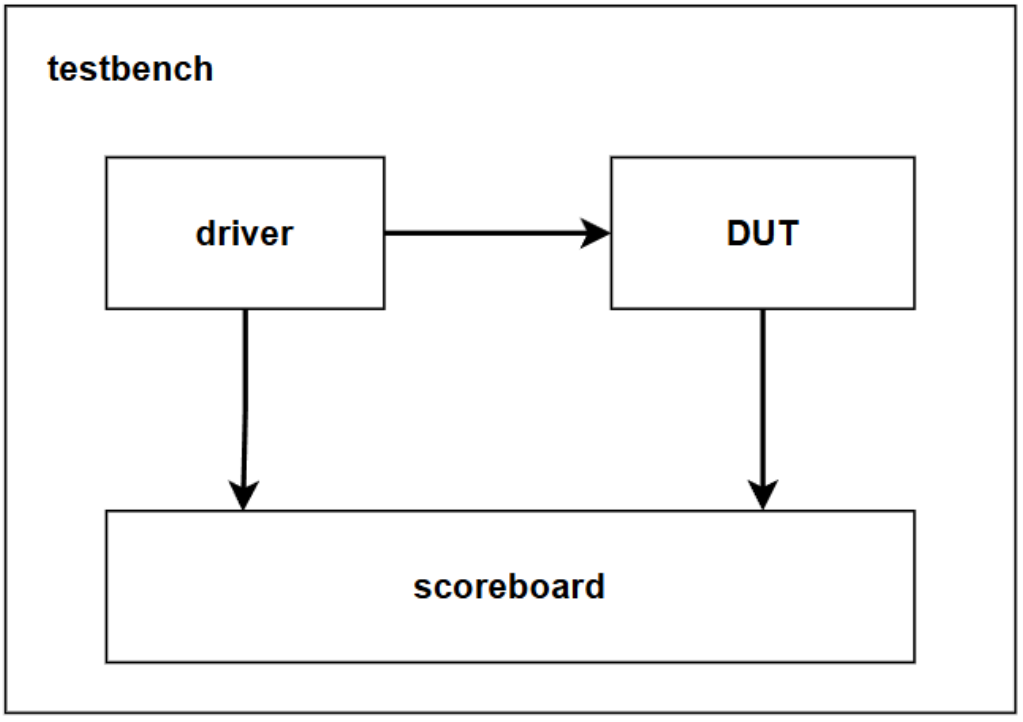
Tệp dùng để test gồm 4 thư mục chính:

- **00_src**: chứa toàn bộ mã nguồn RTL (source code) của các bạn.
- **01_bench**: chứa tất cả các file testbench.
- **02_test**: chứa các chương trình test (test program) dùng để kiểm tra hoạt động của DUT.
- **03_sim**: chứa file makefile và các file phục vụ cho việc mô phỏng.

Trong cấu trúc testbench cơ bản, có 3 thành phần chính là **driver**, **DUT (Design Under Test)** và **scoreboard**.

- **Driver** chịu trách nhiệm tạo và gửi các tín hiệu đầu vào đến DUT, đồng thời gửi thông tin tương tự đến **scoreboard** để làm dữ liệu đối chiếu.
- **DUT** là module mà bạn muốn test. Trong quá trình hoạt động, DUT sẽ tạo ra các tín hiệu đầu ra, những tín hiệu này được chuyển đến **scoreboard** để so sánh với kết quả mong đợi.
- **Scoreboard** sẽ kiểm tra kết quả DUT trả về xem có đúng với giá trị kỳ vọng không. Nếu DUT vượt qua tất cả các bài test, quá trình kiểm thử được xem là thành công.

Kết quả của từng bài test sẽ được hiển thị trực tiếp trên **Terminal** sau khi mô phỏng hoàn tất.



Trong testbench mẫu được cung cấp, có hai tệp chương trình:

- `isa_1b.hex` : mỗi dòng chứa **1 byte dữ liệu**.
- `isa_4b.hex` : mỗi dòng chứa **4 byte dữ liệu**.

Hai tệp này thực chất là **cùng một chương trình**, chỉ khác nhau về **định dạng dữ liệu**, nhằm tạo **sự linh hoạt** cho việc khởi tạo bộ nhớ có **độ rộng 8-bit hoặc 32-bit**.

> **Lưu ý:**
>
> Để chạy được chương trình `isa_1b.hex` hoặc `isa_4b.hex`, bộ nhớ lệnh (Instruction Memory) của bạn cần có dung lượng tối thiểu 8 kB, và bộ nhớ dữ liệu (Data Memory) cần có ít nhất 2 kB.

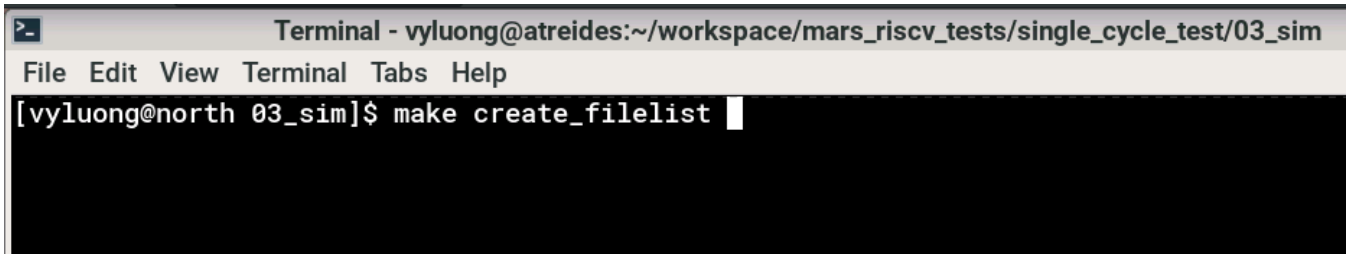
2. Cách chạy simulation

Để thiết lập và chạy mô phỏng, bạn chỉ cần đặt tất cả file code RTL của mình (các file có phần mở rộng `.sv` hoặc `.v`) vào thư mục **00_src**, sau đó vào thư mục **03_sim** và chạy lệnh mô phỏng được định nghĩa trong file *Makefile*.

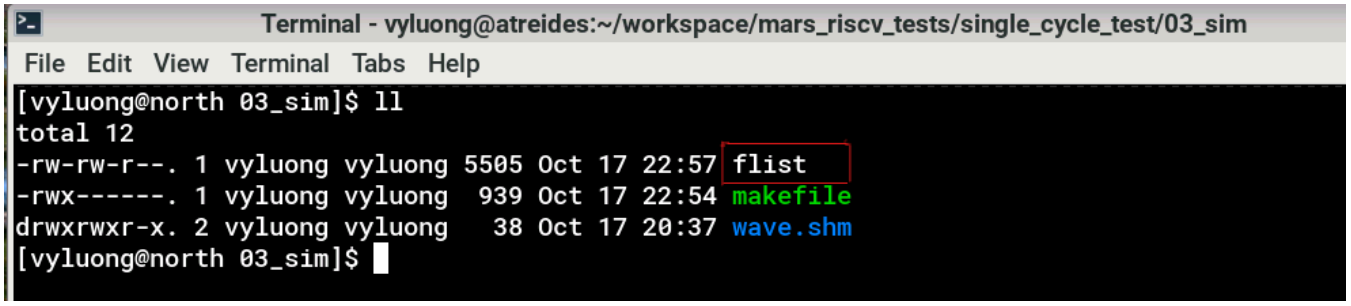
> **Lưu ý: Các bạn không cần chỉnh sửa bất kỳ file testbench nào.**
>
> Toàn bộ testbench đã được chuẩn bị sẵn, chỉ cần đảm bảo rằng bạn tuân thủ đúng hướng dẫn trong milestone 2, bao gồm khai báo đầy đủ các tín hiệu đầu vào, đầu ra và tên tín hiệu theo yêu cầu.

Ban đầu, thư mục **03_sim** chỉ chứa file `makefile`. Bước đầu tiên bạn cần làm là tạo một file có tên **flist** (filelist), trong đó liệt kê tất cả đường dẫn đến các file code RTL và testbench. Trình mô phỏng sẽ dựa vào danh sách này để tìm và biên dịch code RTL.

Trong `makefile`, đã có sẵn một lệnh giúp bạn tạo filelist một cách tự động. Chỉ cần chạy lệnh sau trong terminal:



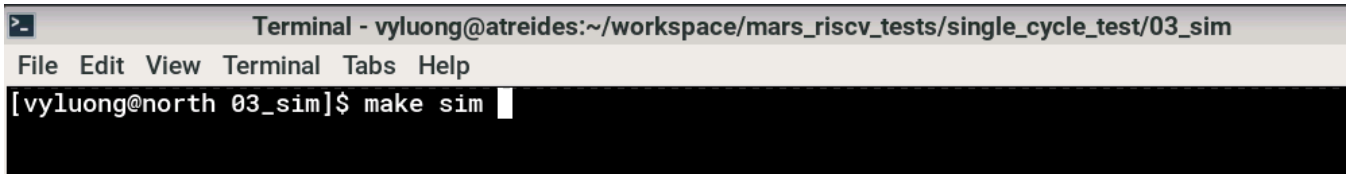
Lệnh này sẽ tự động tìm tất cả các file `.sv`, `.v` và `.svh` trong hai thư mục **00_src** và **01_bench**, sau đó ghi đường dẫn của chúng vào file **flist**. Sau khi chạy xong, bạn có thể dùng lệnh `ls` để kiểm tra xem filelist (`flist`) đã được tạo thành công hay chưa.



> **Ghi chú:**
>
> Lệnh `make create_filelist` giúp gom tất cả các file RTL vào filelist. Tuy nhiên, bạn nên **sắp xếp lại thứ tự trong filelist** sao cho phản ánh đúng **hiearchy (thứ bậc) module trong thiết kế**, tức là các module top level nên nằm ở cuối danh sách, còn các module con ở phía trên.

Trình mô phỏng sẽ đọc file từ trên xuống, vì vậy nếu sắp xếp sai thứ tự, có thể trình mô phỏng sẽ không tìm thấy module quan trọng trước khi cần biên dịch.

Sau khi đã có filelist, bạn có thể bắt đầu chạy mô phỏng bằng lệnh:



Bài test này dùng để kiểm tra chức năng cơ bản của từng lệnh trong tập lệnh RV32I. Khi chạy mô phỏng, **scoreboard** trong testbench sẽ hiển thị **trạng thái của bài test** trên terminal, như hình bên dưới:

```
SINGLE CYCLE - ISA test

add......PASS
addi.....PASS
sub......PASS
and......PASS
andi.....PASS
or.......PASS
ori......PASS
xor......PASS
xori.....PASS
slt......PASS
slti.....PASS
sltu.....PASS
sltiu....PASS
sll......PASS
slli.....PASS
srl......PASS
srli.....PASS
sra......PASS
srai.....PASS
lw.......PASS
lh.......PASS
lhu......PASS
lb.......PASS
lbu......PASS
sw.......PASS
sh.......PASS
sb.......PASS
auipc....PASS
lui......PASS
beq......PASS
bne......PASS
blt......PASS
bltu.....PASS
bge......PASS
bgeu.....PASS
jal......PASS
jalr.....PASS
malgn....PASS
iosw.....PASS

Timeout...


Simulation complete via $finish(1) at time 40 US + 0
../01_bench/tlib.svh:29             $finish;
xcelium> exit
TOOL:   xrun(64)        24.09-s005: Exiting on Oct 20, 2025 at 08:31:40 +07  (total: 00:00:04)
```

⚠ **Lưu ý:**

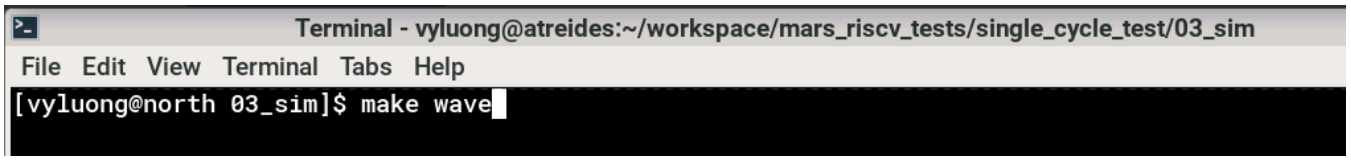
Nếu bạn chạy mô phỏng mà **không thấy trạng thái của bài test xuất hiện**, hoặc **kết quả bị gián đoạn trong thời gian dài**, điều đó có nghĩa là CPU Single Cycle của bạn đang gặp lỗi ở lệnh nhánh (Branch instruction) và có thể khiến lõi bị treo vô hạn.

Trong trường hợp này, bạn cần **kiểm tra lại cách xử lý lệnh nhánh** trong thiết kế của mình trước khi chạy lại bài test.

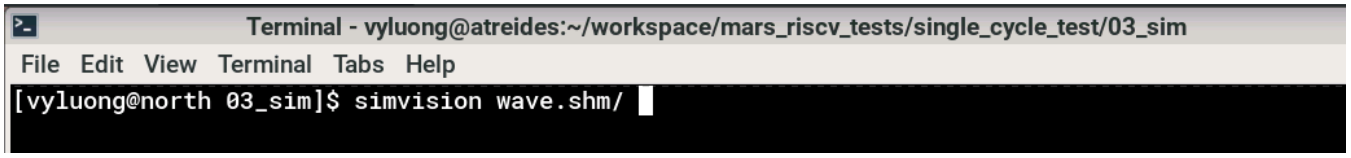
Hãy đảm bảo các bạn thiết lập thời gian timeout phù hợp để chương trình có đủ thời gian chạy toàn bộ các bài kiểm thử.

Sau khi hoàn thành tất cả các bài test, chương trình sẽ kết thúc và nhảy vào một vòng lặp vô hạn (infinite loop subroutine) đẻ không thực thi câu lệnh nào mới.

Nếu trình mô phỏng xuất hiện lỗi hoặc cảnh báo, hãy kiểm tra lại mã RTL của bạn. Khi mô phỏng hoàn tất, bạn có thể quan sát dạng sóng (waveform) để kiểm tra hoạt động của thiết kế bằng một trong hai lệnh sau để mở **SimVision waveform viewer**, giúp bạn dễ dàng quan sát, debug và xác minh hoạt động của thiết kế:



hoặc dùng `simvision` **+** `tên file wave` **:**



3. Cách chấm điểm

Để được chấm điểm, bạn cần **nộp mã RTL và báo cáo** trong một file `.zip` trước hạn nộp (sẽ được thông báo sau).

Trợ giảng sẽ nhận code RTL của bạn, tiến hành chấm điểm tự động (autograding) bằng cách chạy các bài test giống như hướng dẫn này nhằm xác minh lại kết quả thiết kế của bạn.

Ngoài ra, còn có phần báo cáo và vấn đáp, trong đó giảng viên hoặc trợ giảng sẽ xem xét kết quả mô phỏng và đặt câu hỏi để đánh giá mức độ hiểu biết và mức độ hoàn thiện của thiết kế.