**1. Ý nghĩa module**

Đây là **Register File** của vi xử lý RISC-V, nhiệm vụ:

* Chứa **32 thanh ghi, mỗi thanh ghi 32 bit**.
* Cho phép **ghi 1 thanh ghi** (write port) và **đọc đồng thời 2 thanh ghi** (read port A và B).
* Thanh ghi x0 luôn bằng 0.

**2. Cấu trúc module**

Module có 2 chế độ:

1. **SUPPORT\_REGFILE\_XILINX = 1**  
   → sử dụng một module đặc biệt (riscv\_xinlinx\_2r1w) tối ưu cho FPGA Xilinx (tận dụng block RAM dual-port).
2. **SUPPORT\_REGFILE\_XILINX = 0**  
   → dùng mảng các reg trong Verilog (mỗi thanh ghi là một biến reg riêng).  
   → đây là phần bạn đưa chi tiết.

**3. Các cổng tín hiệu**

* **Input**
  + clk\_i: clock.
  + rst\_i: reset đồng bộ.
  + rd0\_i [4:0]: địa chỉ thanh ghi đích cần ghi (rd).
  + rd0\_value\_i [31:0]: dữ liệu cần ghi.
  + ra0\_i [4:0]: địa chỉ thanh ghi nguồn A (rs1).
  + rb0\_i [4:0]: địa chỉ thanh ghi nguồn B (rs2).
* **Output**
  + ra0\_value\_o [31:0]: dữ liệu đọc từ rs1.
  + rb0\_value\_o [31:0]: dữ liệu đọc từ rs2.

**4. Nguyên lý hoạt động**

**a) Bộ nhớ thanh ghi**

* reg\_r1\_q → reg\_r31\_q là 31 thanh ghi thật sự (x1 đến x31).
* x0 không có reg, mà được hardwired = 32'b0.

**b) Ghi dữ liệu (write port)**

Khối always @(posedge clk\_i):

* Nếu rst\_i = 1 → reset tất cả thanh ghi về 0.
* Ngược lại → nếu có địa chỉ ghi rd0\_i (≠ 0) → ghi rd0\_value\_i vào đúng thanh ghi tương ứng (reg\_rX\_q).
* if (rd0\_i == 5'd1) reg\_r1\_q <= rd0\_value\_i;
* ...
* if (rd0\_i == 5'd31) reg\_r31\_q <= rd0\_value\_i;
* Nếu rd0\_i = 0 thì không ghi gì (vì x0 luôn bằng 0).

➡️ **Đảm bảo tính chất RISC-V: x0 = 0, không ghi đè được.**

**c) Đọc dữ liệu (read port)**

Khối always @(\*) với case cho ra0\_i và rb0\_i:

* Nếu ra0\_i = 0 → trả về 32'h00000000.
* Nếu ra0\_i = k (1..31) → trả về reg\_rk\_q.
* Tương tự cho rb0\_i.

➡️ Đây là **combinational read**: dữ liệu xuất ra ngay khi thay đổi địa chỉ.

**d) Giao tiếp với pipeline**

* Ở stage **Decode (ID)**: CPU đưa rs1, rs2 vào ra0\_i, rb0\_i → RF trả dữ liệu ra ra0\_value\_o, rb0\_value\_o.
* Ở stage **Writeback (WB)**: CPU đưa rd vào rd0\_i, và kết quả vào rd0\_value\_i, we implicit (ở đây thiết kế luôn ghi nếu rd != 0).

**5. Chức năng bổ sung (cho mô phỏng)**

Khi biên dịch với **Verilator**, có thêm 2 function:

* get\_register(r) → đọc giá trị của thanh ghi r.
* set\_register(r, value) → gán giá trị cho thanh ghi r.

➡️ Giúp **debug, testbench, waveform** dễ theo dõi.

**6. Tóm tắt**

* **Reset** → tất cả thanh ghi x1–x31 = 0.
* **Ghi** → ở cạnh dương clock, nếu rd0\_i != 0, ghi rd0\_value\_i vào thanh ghi đó.
* **Đọc** → combinational, ra0\_value\_o = giá trị tại ra0\_i, rb0\_value\_o = giá trị tại rb0\_i.
* **x0 luôn bằng 0**, bất chấp ghi gì vào.
* Hỗ trợ cả **FPGA tối ưu (Xilinx RAM)** hoặc **mảng reg mô phỏng**.