Dual-Port RAM (with collision flag)

- A small synchronous dual-port RAM allowing 2 independent port (A and B) to read and write the same 4x4 memory concurrently.
- Each port has its own addr, data-in, read/write controls.
- Similar to normal RAM, on en=1 and R_W=1 the i/p data stored in memory and R_W=0 the stored data in memory goes as o/p.
- Collision flag :- If both ports write to the same addr. in same clk., the module asserts a collision flag. o/p drive Z during writes (behav. bus release)

- Circuit Explainat^n

① storage & init

=> register[0..3] holds 4 bit words

=> initial sets data_out_A/B = Z, and collision = 0 along with zeroes memory.

② Port A logic

```
always @(posedge clk) begin
  if(R_W_A) begin
    register[address_A]<= data_in_A;
    data_out_A <= 4'dz;
  end else
    data_out_A <= register [address_A];
end
```

- on clk :- if R_w_A = 1 (write)
  data_in_A to reg [address_A]
  and data_out_A to Z
  else drive data_out with reg [addr_A]
  or the read op^n

③ **Port B logic**

- identical behav. for port B with its own signals.

```
// collision detectⁿ & duplicate writes
always @(posedge clk) begin
  collision <= 1'b0;
   if (R_w_A && Rw_B && address_A ==
                               address_B)
  begin
    collision <= 1'b1;
  end
  else begin
     if (R_w_A) register[address_A] <= data_in_A;
     if (R_w_B) register[address_B] <= data_in_B;
  end
end
```

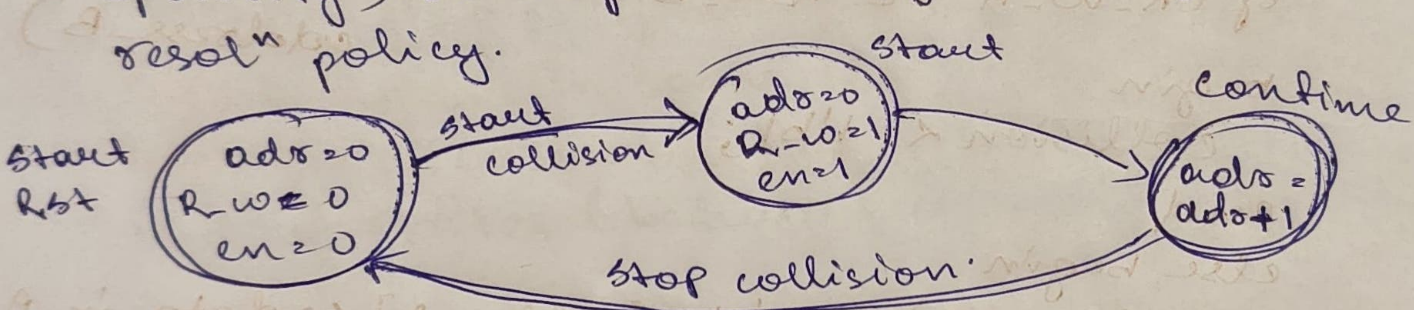
- Clears collision, then if both ports are writing the same addr sets collision = 1
- Else it performs write for any port that requests it.

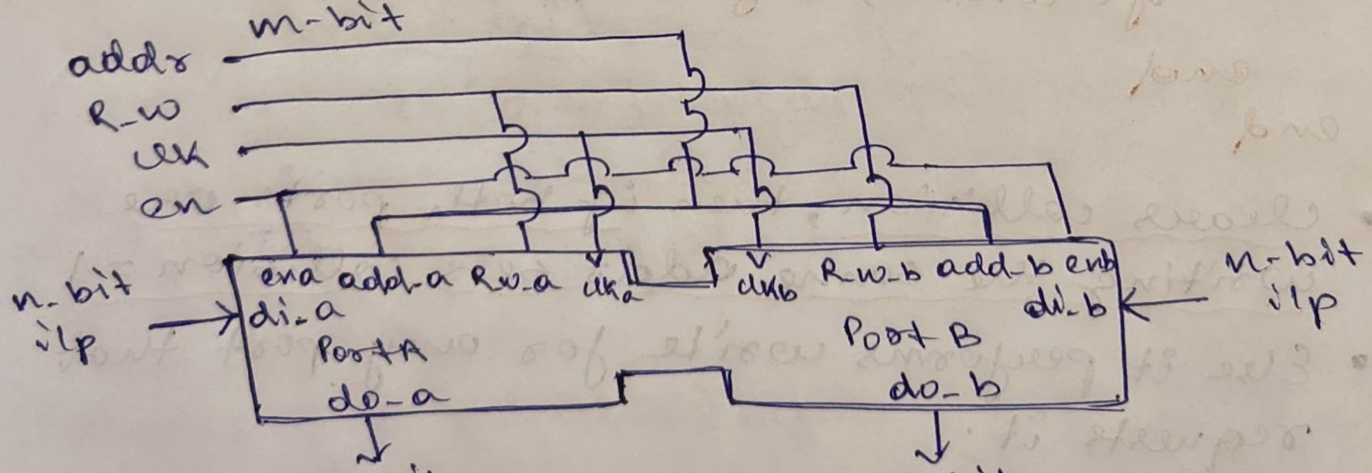
- Summary :-

=> The module intends dual-port behaviours and collision reporting, but the duplicated write logic (per-port blocks + collision block) is redundant and can cause multiple conflicting assign, consolidate writes into a single clocked block to avoid races.

=> o/ps driven to z during write for synthesis prefer defined o/p behaviour or tri-state only on top-level I/O with proper bus arbitratⁿ

=> collision flag is useful - but productⁿ design should define which write wins (priority) or implement byte-merge / resolⁿ policy.


start
Rst
adr=0
R_w=0
en=0
start
collision
start
adr=0
R_w=1
en=1
continue
adr = adr+1
stop collision


m-bit
addr
R_w
clk
en
n-bit
i/p
ena
add_a
Rw_a
clk_a
di_a
Port A
do_a
n-bit
clkb
R_w_b
add_b
enb
di_b
Port B
do_b
n-bit
n-bit
i/p