DESIGN:

```
module mul_mem (
input logic [3:0] a,
input logic [3:0] b,
output logic [7:0] product, //result of 4x4 is 8bit
input logic clk,rst,
input logic  wr_en,rd_en, //write and read enable
input logic [3:0] addrs, //address register
output logic [7:0] data
);

logic [7:0] memory [15:0]; // 16x8 bit mem
logic [7:0] temp;
logic [3:0] count;


always_ff @(posedge clk) begin
if(rst) //assumption we should begin with reset
begin
count <= 0;
temp <= 0;
//addrs = {b,a}; //randomnly address generation
memory[addrs] <= {a,b};
end

else if(wr_en) begin
if (count < b) begin
temp <= temp + a;
count <= count +1;
end
else begin
//addrs <= addrs+incr;
memory[addrs] <= temp; //actual product
product <= temp;
end
end
end

always_ff @(posedge clk)
if(rd_en && !wr_en)
begin
 product <= memory[addrs];
end
```

```
assign data = memory[addrs];

endmodule


/***********************************************/
```

TB:

```
module tb;
 logic [3:0] a;
 logic [3:0] b;
 logic [7:0] product;
 logic clk,rst;
 logic  wr_en,rd_en;
 logic [3:0] addrs;
 logic [7:0] data;

//port mapping

mul_mem dut(.*);

initial begin
clk=0;
rst=1;
product=0;
wr_en=0;
rd_en=0;
a = 4'b0110;
b = 4'b0010;
addrs=$random();
#20;

rst=0;
wr_en=1;
addrs=addrs+1; //increment the address
#50;
rst=1;
wr_en=0;
#10;

rst=0;
rd_en=1;
#50;
end
initial #150 $finish;
always #5 clk = ~clk;
endmodule
```

Output Waveform: