

```
`include "timescale.v"
```

```
module mixcolumn(clk,reset,decrypt_i,start_i,data_i,ready_o,data_o);
```

```
input clk;
```

```
input reset;
```

```
input decrypt_i;
```

```
input start_i;
```

```
input [127:0] data_i;
```

```
output ready_o;
```

```
output [127:0] data_o;
```

```
reg ready_o;
```

```
reg [127:0] data_o;
```

```
reg [127:0] data_reg;
```

```
reg [127:0] next_data_reg;
```

```
reg [127:0] data_o_reg;
```

```
reg [127:0] next_data_o;
```

```
reg next_ready_o;
```

```
reg [1:0] state;
```

```
reg [1:0] next_state;
```

```
wire [31:0] outx;
```

```
wire [31:0] outy;
```

```
reg [31:0] mix_word;
```

```
reg [31:0] outmux;
```

```
word_mixcolumn w1 (.in(mix_word), .outx(outx), .outy(outy));
```

```
//assign_data_o:
```

```
always @(data_o_reg)
```

```
begin
```

```
    data_o = (data_o_reg);
```

```
end
```

```
//mux:
```

```
always @(outx or outy or decrypt_i)
```

```
begin
```

```
    outmux = (decrypt_i?outy:outx);
```

```
end
```

```
//registers:
```

```
always @(posedge clk or negedge reset)
```

```
begin
```

```
    if(!reset)
```

```
    begin
```

```
        data_reg = (0);
```

```
        state = (0);
```

```
        ready_o = (0);
```

```
    data_o_reg = (0);  
end  
else  
begin  
  
    data_reg = (next_data_reg);  
    state = (next_state);  
    ready_o = (next_ready_o);  
    data_o_reg = (next_data_o);  
  
end  
  
end  
  
//mixcol:  
reg[127:0] data_i_var;  
reg[31:0] aux;  
reg[127:0] data_reg_var;
```

```
always @(decrypt_i or start_i or state or data_reg or outmux or data_o_reg or data_i)
```

```
begin
```

```
    data_i_var=data_i;  
    data_reg_var=data_reg;  
    next_data_reg = (data_reg);  
    next_state = (state);  
  
    mix_word = (0);
```

```
next_ready_o = (0);
```

```
next_data_o = (data_o_reg);
```

```
case(state)
```

```
0:
```

```
begin
```

```
if(start_i)
```

```
begin
```

```
aux=data_i_var[127:96];
```

```
mix_word = (aux);
```

```
data_reg_var[127:96]=outmux;
```

```
next_data_reg = (data_reg_var);
```

```
next_state = (1);
```

```
end
```

```
end
```

```
1:
```

```
begin
```

```
aux=data_i_var[95:64];
```

```
mix_word = (aux);
```

```
data_reg_var[95:64]=outmux;
```

```
next_data_reg = (data_reg_var);
```

```
next_state = (2);
```

```
end
```

```
2:
```

```
begin
```

```
aux=data_i_var[63:32];
```

```
mix_word = (aux);
```

```
    data_reg_var[63:32]=outmux;
    next_data_reg = (data_reg_var);
    next_state = (3);
end
3:
begin
    aux=data_i_var[31:0];
    mix_word = (aux);
    data_reg_var[31:0]=outmux;
    next_data_o = (data_reg_var);
    next_ready_o = (1);
    next_state = (0);
end
default:
begin
end

endcase

end

endmodule
```