

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

```
module  
keysched(clk,reset,start_i,round_i,last_key_i,new_key_o,ready_o,sbox_access_o,sbox_data_o,sbox_data_i,sbox_decrypt_o);
```

```
input clk;
```

```
input reset;
```

```
input start_i;
```

```
input [3:0] round_i;
```

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

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

```
output ready_o;
```

```
output sbox_access_o;
```

```
output [7:0] sbox_data_o;
```

```
input [7:0] sbox_data_i;
```

```
output sbox_decrypt_o;
```

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

```
reg ready_o;
```

```
reg sbox_access_o;
```

```
reg [7:0] sbox_data_o;
```

```
reg sbox_decrypt_o;
```

```
reg [2:0] next_state;
```

```
reg [2:0] state;
```

```
reg [7:0] rcon_o;
```

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

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

```
reg [127:0] key_reg;  
reg [127:0] next_key_reg;  
reg next_ready_o;
```

```
//rcon:
```

```
always @( round_i)
```

```
begin
```

```
case(round_i)
```

```
1:
```

```
begin
```

```
    rcon_o = (1);
```

```
end
```

```
2:
```

```
begin
```

```
    rcon_o = (2);
```

```
end
```

```
3:
```

```
begin
```

```
    rcon_o = (4);
```

```
end
```

```
4:
```

```
begin
```

```
    rcon_o = (8);
```

```
end
```

```
5:
```

```
begin
```

```
    rcon_o = ('h10');  
end  
6:  
begin  
    rcon_o = ('h20');  
end  
7:  
begin  
    rcon_o = ('h40');  
end  
8:  
begin  
    rcon_o = ('h80');  
end  
9:  
begin  
    rcon_o = ('h1B');  
end  
10:  
begin  
    rcon_o = ('h36');  
end  
default:  
begin  
    rcon_o = (0);  
end  
endcase  
  
end
```

```
//registers:
always @(posedge clk or negedge reset)
begin

    if(!reset)
        begin

            state = (0);
            col = (0);
            key_reg = (0);
            ready_o = (0);

        end
    else
        begin

            state = (next_state);
            col = (next_col);
            key_reg = (next_key_reg);
            ready_o = (next_ready_o);

        end

    end

end

//generate_key:
```

```
reg[127:0] K_var,W_var;
```

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

```
reg[23:0] zero;
```

```
always @(start_i or last_key_i or sbbox_data_i or state or rcon_o or col or key_reg)
```

```
begin
```

```
    zero=0;
```

```
    col_t=col;
```

```
    W_var=0;
```

```
    next_state = (state);
```

```
    next_col = (col);
```

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

```
    next_key_reg = (key_reg);
```

```
    new_key_o = (key_reg);
```

```
    sbbox_decrypt_o = (0);
```

```
    sbbox_access_o = (0);
```

```
    sbbox_data_o = (0);
```

```
    K_var=last_key_i;
```

```
    case(state)
```

```
        //Substitute the bytes while rotating them
```

```
        //Four accesses to SBox are needed
```

```
    0:
```

```
begin
    if(start_i)
        begin

            col_t=0;
            sbox_access_o = (1);
            sbox_data_o = (K_var[31:24]);
            next_state = (1);

        end
    end
```

```
1:
    begin

        sbox_access_o = (1);
        sbox_data_o = (K_var[23:16]);
        col_t[7:0]=sbox_data_i;
        next_col = (col_t);
        next_state = (2);

    end
```

```
2:
    begin

        sbox_access_o = (1);
        sbox_data_o = (K_var[15:8]);
        col_t[31:24]=sbox_data_i;
```

```
next_col = (col_t);
```

```
next_state = (3);
```

```
end
```

```
3:
```

```
begin
```

```
sbox_access_o = (1);
```

```
sbox_data_o = (K_var[7:0]);
```

```
col_t[23:16]=sbox_data_i;
```

```
next_col = (col_t);
```

```
next_state = (4);
```

```
end
```

```
4:
```

```
begin
```

```
sbox_access_o = (1);
```

```
col_t[15:8]=sbox_data_i;
```

```
next_col = (col_t);
```

```
W_var[127:96]=col_t^K_var[127:96]^{rcon_o,zero};
```

```
W_var[95:64]=W_var[127:96]^K_var[95:64];
```

```
W_var[63:32]=W_var[95:64]^K_var[63:32];
```

```
W_var[31:0]=W_var[63:32]^K_var[31:0];
```

```
next_ready_o = (1);
```

```
next_key_reg = (W_var);
```

```
    next_state = (0);
```

```
end
```

```
default:
```

```
    begin
```

```
        next_state = (0);
```

```
    end
```

```
endcase
```

```
end
```

```
endmodule
```