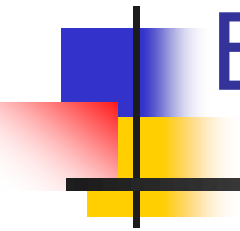
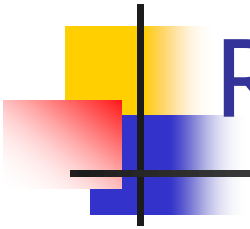


EL6463: Round Key Generation



RC5 Key Expansion

- 128-bit user key->26 round keys (each 32-bit)
  - User Key->S[0], S[1]…, S[25]

Entity definition

```
ENTITY key_exp IS
    PORT
    (
        clr, clk        : STD_LOGIC;  -- by default, it is input
        key_in          : STD_LOGIC;
        ukey            : STD_LOGIC_VECTOR(127 DOWNTO 0);
        skey            : OUT  STD_LOGIC_VECTOR(31 DOWNTO 0);
        key_rdy         : OUT  STD_LOGIC
    );
END key_exp;
```

Signal declarations

```
SIGNAL  a_reg      : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  b_reg      : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  a_tmp1     : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  a_tmp2     : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  ab_tmp     : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  b_tmp1     : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  b_tmp2     : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL  l_arr      : L_ARRAY;
SIGNAL  s_arr_tmp: S_ARRAY;

SIGNAL  i_cnt      : INTEGER RANGE 0 TO 25;  -- you can use std_logic_vector
SIGNAL  j_cnt      : INTEGER RANGE 0 TO 3;  -- you can use std_logic_vector
SIGNAL  k_cnt      : INTEGER RANGE 0 TO 77; -- any better way?
```

Use of VHDL package

- The definitions and declarations in a package can be shared by multiple VHDL models

```
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     WORK.RC5_PKG.ALL;  -- include the package to your design
……
```

```
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;

PACKAGE rc5_pkg IS
  TYPE   S_ARRAY IS ARRAY (0 TO 25) OF STD_LOGIC_VECTOR(31 DOWNTO 0);
  TYPE   L_ARRAY IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(31 DOWNTO 0);
END rc5_pkg;
```

Round Key Generation

```
do 3*max(t, c); t=26, c=4
        A = S[i] = (S[i] + A + B) <<< 3;
        B = L[j] = (L[j] + A + B) <<< (A + B);
        i = (i + 1) mod (t);
        j = (j + 1) mod (c);
```

```
i=0;j=0;
do 78 times
        {
                A = S[i] = (S[i] + A + B) <<< 3;
                B = L[j] = (L[j] + A + B) <<< (A + B);
                i = (i + 1) mod 26;
                j = (j + 1) mod 4;
        }
```

Round key generation: Operation 1

```
-- it is not a data-dependent rotation!
A = S[i] = (S[i] + A + B) <<< 3;
```

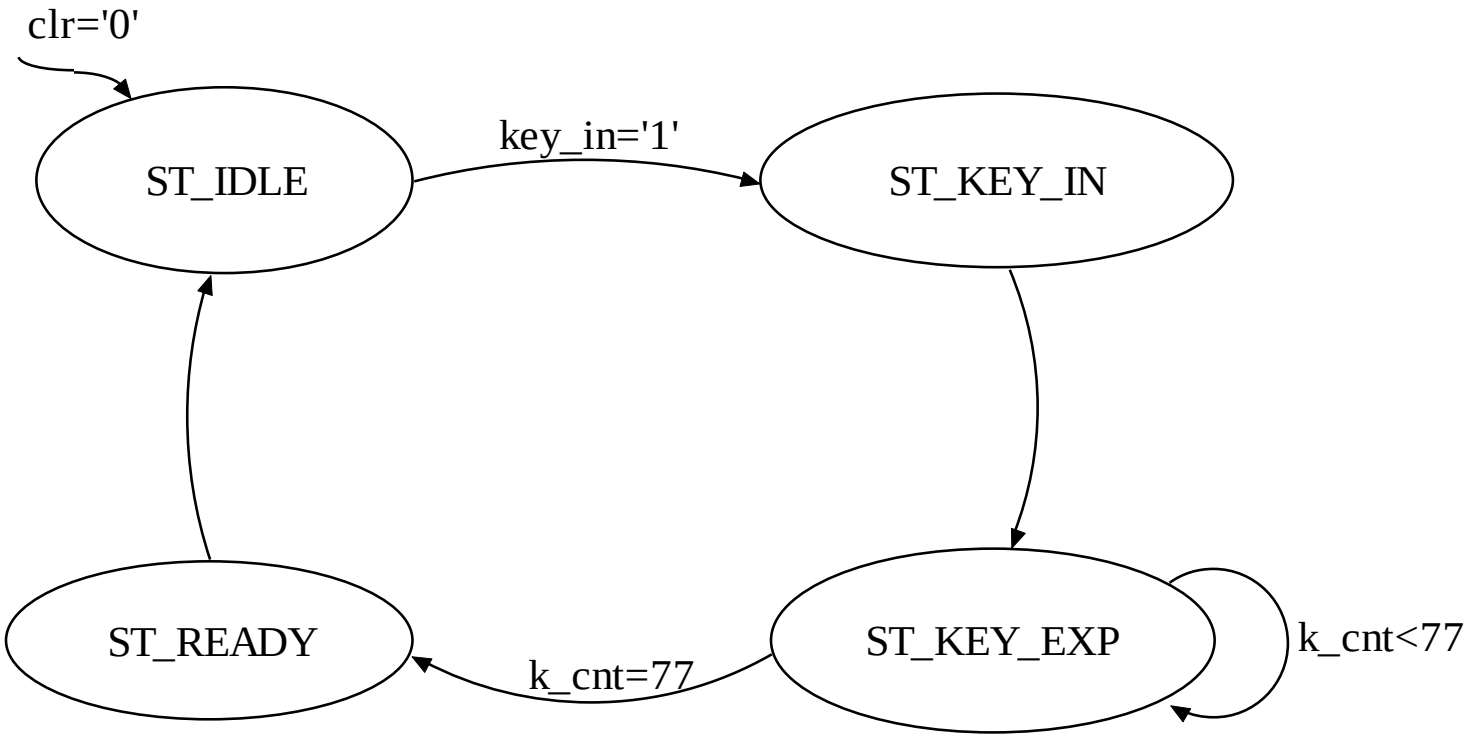
```
a_tmp1<=s_arr_tmp(i_cnt)+a_reg+b_reg; -- not a good style!
-- <<<3
a_tmp2<=a_tmp1(28 DOWNTO 0) & a_tmp1(31 DOWNTO 29);
```

Round key generation: Operation 2

```
-- this is a data-dependent rotation!
B = L[j] = (L[j] + A + B) <<< (A + B);
```

```
ab_tmp<=a_tmp2+b_reg;
b_tmp1<=l_arr(j_cnt)+ab_tmp;
WITH ab_tmp(4 DOWNTO 0) SELECT
  b_tmp2<=
   b_tmp1(30 DOWNTO 0) & b_tmp1(31) WHEN "00001",
   b_tmp1(29 DOWNTO 0) & b_tmp1(31 DOWNTO 30) WHEN "00010",
   b_tmp1(28 DOWNTO 0) & b_tmp1(31 DOWNTO 29) WHEN "00011",
   ……
   b_tmp1(0) & b_tmp1(31 DOWNTO 1)   WHEN "11111",
   b_tmp1   WHEN OTHERS;
```

Round key generation: state m/c



Round key generation: state m/c

```
TYPE     StateType IS (ST_IDLE, ST_KEY_INIT, ST_KEY_EXP, ST_READY);
SIGNAL  state : StateType;
……
  PROCESS(clr, clk)
     BEGIN
       IF(clr='0') THEN
           state<=ST_IDLE;
       ELSIF(clk'EVENT AND clk='1') THEN
           CASE state IS
              WHEN ST_IDLE | ST_READY=>
                  IF(key_in='1') THEN  state<=ST_KEY_INIT;   END IF;
              WHEN ST_KEY_INIT=>
                  state<=ST_KEY_EXP;
              WHEN ST_KEY_EXP=>
                  IF(k_cnt=77) THEN   state<=ST_READY;  END IF;
          END CASE;
        END IF;
  END PROCESS;
```

A register and B register

```
-- A register
PROCESS(clr, clk)  BEGIN
    IF(clr='0') THEN
       a_reg<=(OTHERS=>’0’);
    ELSIF(clk'EVENT AND clk='1') THEN
       IF(state=ST_KEY_EXP) THEN   a_reg<=a_tmp2;
       END IF;
    END IF;
END PROCESS;
-- B register
PROCESS(clr, clk)  BEGIN
    IF(clr='0') THEN
       b_reg<=(OTHERS=>’0’);
    ELSIF(clk'EVENT AND clk='1') THEN
       IF(state=ST_KEY_EXP) THEN   b_reg<=b_tmp2;
       END IF;
    END IF;
END PROCESS;
```

Two arrays: S_ARRAY, L_ARRAY

- and two counters
  - i_cnt for S_ARRAY, j_cnt for L_ARRAY

```
i = (i + 1) mod (t);
```

```
PROCESS(clr, clk)
BEGIN
   IF(clr='0') THEN  i_cnt<=0;
   ELSIF(clk'EVENT AND clk='1') THEN
      IF(state=ST_KEY_EXP) THEN
        IF(i_cnt=25) THEN   i_cnt<=0;
        ELSE   i_cnt<=i_cnt+1;
        END IF;
      END IF;
   END IF;
END PROCESS;
```

```
j = (j + 1) mod (c);
```

```
PROCESS(clr, clk)
BEGIN
   IF(clr='0') THEN  j_cnt<=0;
   ELSIF(clk'EVENT AND clk='1') THEN
      IF(state=ST_KEY_EXP) THEN
        IF(j_cnt=3) THEN   j_cnt<=0;
        ELSE   j_cnt<=j_cnt+1;
        END IF;
      END IF;
   END IF;
END PROCESS;
```

Initialize S_ARRAY

S[0] = 0xB7E15163 (*P*w)
for i=1 to 25 do S[i] = S[i-1]+ 0x9E3779B9 (*Q*w)

```
PROCESS(clr, clk)
BEGIN
  IF(clr='0') THEN  -- After system reset, S array is initialized with P and Q
     s_arr_tmp(0)<=  "10110111111000010101000101100011"; Pw
     s_arr_tmp(1)<=  "01010110000110001100101100011100"; Pw+ Qw
     s_arr_tmp(2)<=  "11110100010100000100010011010101"; Pw+ 2Qw
     ……
     s_arr_tmp(25)<="00101011010011000011010001110100"; Pw+ 25Qw
  ELSIF(clk'EVENT AND clk='1') THEN
    IF(state=ST_KEY_EXP) THEN   s_arr_tmp(i_cnt)<=a_tmp2;
    END IF;
  END IF;
END PROCESS;
skey<=s_arr_tmp(25);
```

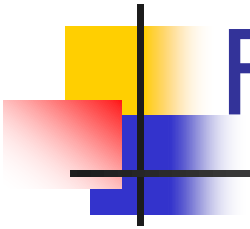
L_ARRAY

```
PROCESS(clr, clk)
BEGIN
  IF(clr= ‘0’ ) THEN
    FOR i IN 0 TO 3 LOOP
      l_arr(i)<=(OTHERS=>'0');
    END LOOP;
  ELSIF(clk'EVENT AND clk='1') THEN
    IF(state=ST_KEY_INIT) THEN
      l_arr(0)<=ukey(31 DOWNTO 0);
      l_arr(1)<=ukey(63 DOWNTO 32);
      l_arr(2)<=ukey(95 DOWNTO 64);
      l_arr(3)<=ukey(127 DOWNTO 96);
    ELSIF(state=ST_KEY_EXP) THEN
      l_arr(j_cnt)<=b_tmp2;
    END IF;
  END IF;
END PROCESS;
```

```
for i = b - 1 downto 0 do
    L[i/u] = (L[i/u] <<< 8) + K[i];
```

Exercise

- Finish RC5 key expansion model
- Simulate RC5 key expansion VHDL model
  - Functional Simulation using ModelSim
  - Code debugging
- Synthesize the Design
  - Perform Timing Simulation
- Understand following concepts/operations
  - Package
  - Data-independent rotate
  - How to do initialization?



Functional Simulation

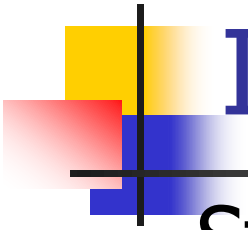
- Create project folder
  - C:\ee4313\proj4
- Save your VHDL code in project folder
  - rc5_pkg.vhd
  - key_exp.vhd
- Start ModelSim

Functional Simulation

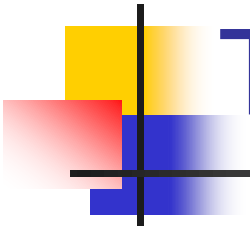
- In ModelSim console window, type following commands in sequence
- With the given ukey (all 0’s), you should get the same skey (declared as ROM) for previous labs

```
cd {c:/ee4313/proj4}
vlib work
vcom rc5_pkg.vhd
vcom key_exp.vhd
vsim key_exp
view wave
add wave clr clk key_in ukey a_reg b_reg skey key_rdy
force clr 0 0, 1 200
force clk 0 0, 1 50 –repeat 100
force ukey 16#00000000000000000000000000000000 0
run 1500
```

FPGA Synthesis and Implementation



- Start Xilinx Project Navigator
- Create new project
  - Project name: key_exp
  - Device: Virtex->xcv1000->bg560->-4
- Create timing simulation model
  - key_exp_sim.vhd



Timing Simulation

- Make sure timing simulation models are successfully generated
  - key_exp_sim.vhd and key_exp_sim.sdf
- Start ModelSim

Timing Simulation

- In ModelSim console window, type in following commands in sequence

```
cd {c:/ee4313/proj4}
vlib work
vmap simprim c:/modeltech_5.6d/xilinx/simprim
vcom key_exp_sim.vhd
vsim -sdftyp /=c:/ee4313/proj4/key_exp_sim.sdf work.key_exp
view wave
add wave clr clk key_in ukey a_reg b_reg skey key_rdy
force clr 0 0, 1 200
force clk 0 0, 1 50 –repeat 100
force ukey 16#00000000000000000000000000000000 0
run 1500
```