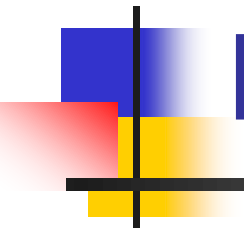
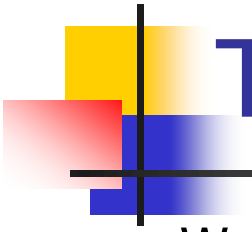


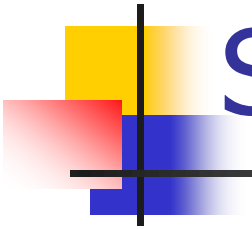
# EL6463 – Structural Modeling

- Structural Modeling

# A Complete Design with All Three Modules



- We designed three modules
  - Key expansion, encryption, decryption
- Put them together to implement RC5
  - Now that we have completed RC5 encryption, RC5 decryption and Round key generation we can combine them together to have a complete working design.
  - To easily connect the RC5 Encryption algorithm with the Round key Generation, we will need to bring out the skey to the top of the encryption algorithm and modify state machine to wait until it receives a valid skey (from the round key generation) before proceeding with encryption.

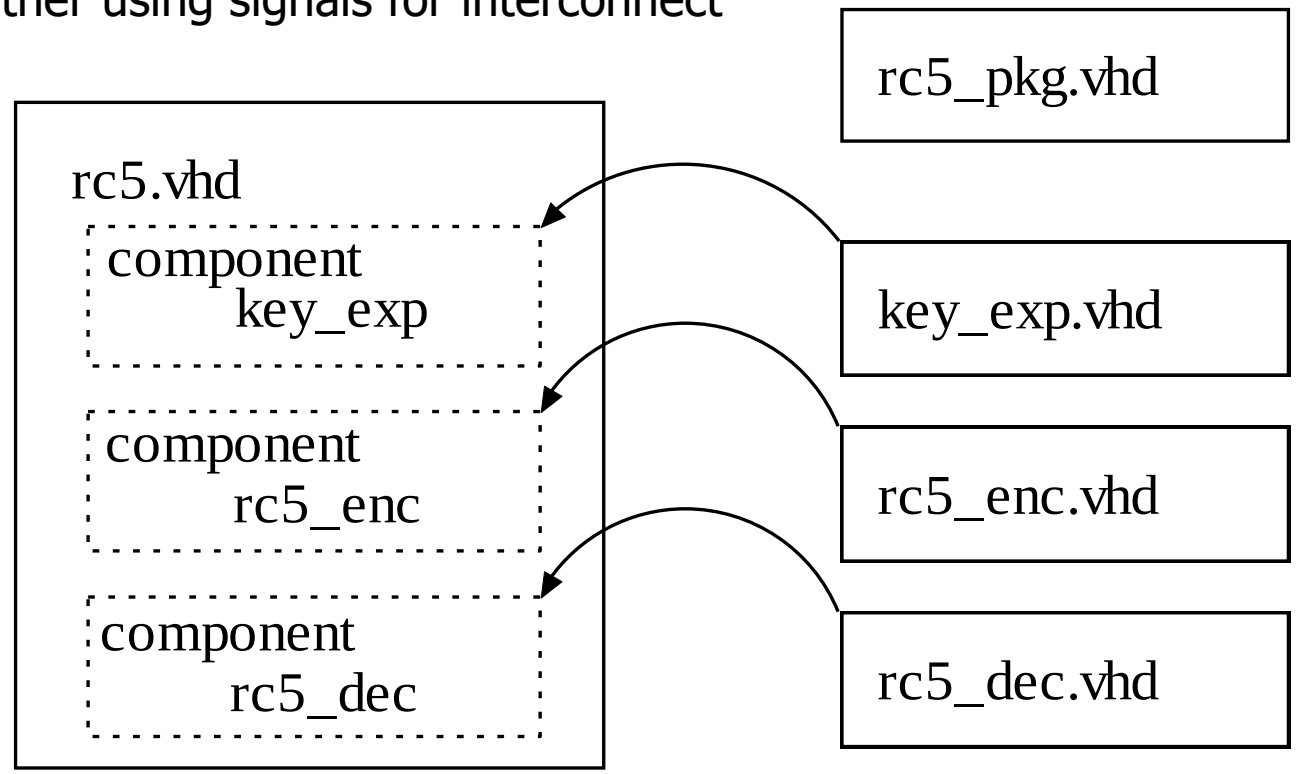


# Structural Description

- If lower level modules are all available, we can assemble them together at a higher level
  - We define entities, then instantiate them as components
- Use wires to interconnect  components
- Three concepts
  - Define components: in a separate VHDL file
  - Declare components: before the architecture
  - Instantiate components: inside architecture

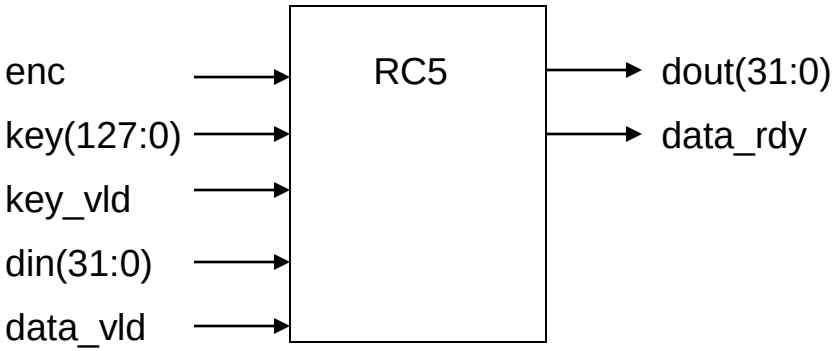
# Instantiate Components

- Map the components into a higher-level design
- To connect all the blocks together we will create an RC5 vhdl file and declare three components (Encryption, Decryption, Round Key) and connect them together using signals for interconnect



# Complete RC5 Port Map

- A top level block diagram for the RC5 design is shown below
- The enc input is used to change between encryption or decryption. The key_vld is used for when the key is valid and data_vld is used for when the data is valid.



# Modifications to RC5 Encryption

- To easily connect the RC5 Encryption algorithm with the Round key Generation, we will need to bring out the skey to the top of the encryption algorithm and modify state machine to wait until it receives a valid skey (from the round key generation) before proceeding with encryption.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
USE            WORK.rc5_pkg.ALL;
entity rc5_enc_full is
               port ( clr, clk: in   std_logic;
                      din:      in   std_logic_vector(63 downto 0);
                      dout:     out  std_logic_vector(63 downto 0);
                      di_vld:   in   std_logic;
                      key_rdy:  in   std_logic;
                      skey:     in   rom;
                      do_rdy:   out  std_logic);

end rc5_enc_full;
```

```
--RC5 state m/c
process(clr, clk)
begin
   if(clr = '0') then
                state<= ST_IDLE;
   elsif (clk'EVENT and clk='1') then
                case state is
                  when ST_IDLE =>
                      if(di_vld='1' and key_rdy='1') then
                         state <= ST_PRE_ROUND;
                      end if;
                  when ST_PRE_ROUND =>
                       state <= ST_ROUND_OP;
                  when ST_ROUND_OP =>
                       if (i_cnt="1100") then
                          state <= ST_READY;
                       end if;
                   when ST_READY =>
                       state <= ST_IDLE;
                end case;
   end if;
end process;
```

# Define Entity

```
ENTITY rc5 IS
   PORT
   (
      clr, clk          : IN    STD_LOGIC;
      enc               : IN    STD_LOGIC;  -- Encryption or decryption?
      key_vld           : IN    STD_LOGIC;  -- Indicate the input is user key
      data_vld          : IN    STD_LOGIC;  -- Indicate the input is user data
      din               : IN    STD_LOGIC_VECTOR(63 downto 0);
      dout              : OUT   STD_LOGIC_VECTOR(63 downto 0);
      data_rdy          : OUT   STD_LOGIC   -- Indicate the output data is ready
   );
END rc5;
```

# Declare Components

- You can declare components in the same design file or in a separate package file

```
-- End  of ENTITY
  -- Internal signal definitions
  -- Component declaration
-- Start of ARCHITECTURE
```

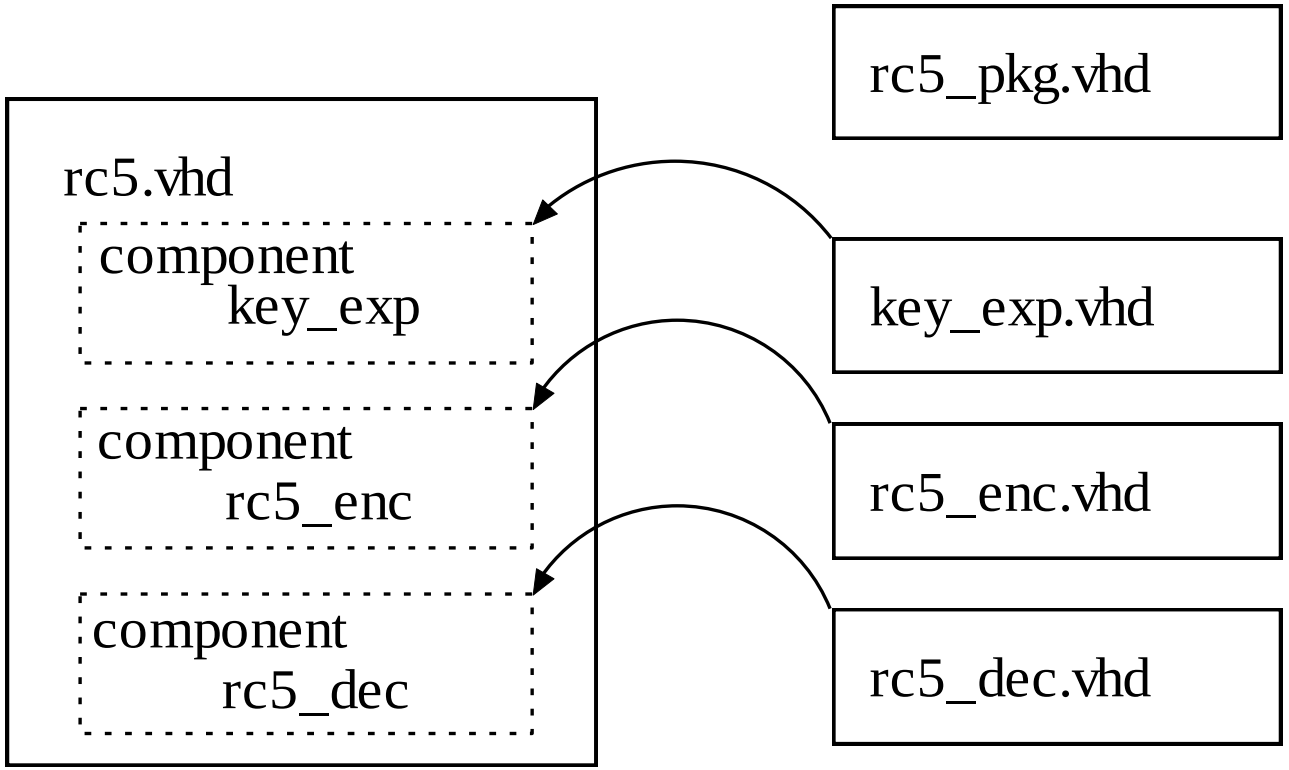
```
.....
PACKAGE rc5_pkg IS
  -- Type definitions for S_ARRAY and L_ARRAY
  -- Component declaration
 COMPONENT  rc5_key  -- Key expansion module
   PORT(
      clr, clk          : IN    STD_LOGIC;
      key_vld           : IN    STD_LOGIC;
      key               : IN    STD_LOGIC_VECTOR(127 DOWNTO 0);
      skey              : OUT   ROM;
```

# Declare Components

```
   key_rdy          : OUT  STD_LOGIC);
END COMPONENT;

COMPONENT  rc5_enc  -- Encryption module
   PORT(
      clr, clk      : IN   STD_LOGIC;
      key_rdy       : IN   STD_LOGIC;
      di_vld        : IN   STD_LOGIC
      din           : IN   STD_LOGIC_VECTOR(63 DOWNTO 0);
      skey          : IN   ROM;
      dout          : OUT  STD_LOGIC_VECTOR(63 DOWNTO 0);
      do_rdy        : OUT  STD_LOGIC)
END COMPONENT;

COMPONENT  rc5_dec  -- Decryption module
   ……
END rc5_pkg;
```

# Instantiate Components

- Map the components into a higher-level design



# Instantiate Components

- Instantiate components in the architecture

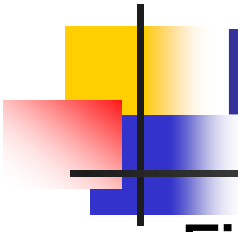
```
ARCHITECTURE struct OF rc5 IS  -- Structural description
    signal skey: rom;
    signal key_rdy: std_logic;
    signal dout_enc: std_logic_vector(63 downto 0);
    signal dout_dec: std_logic_vector(63 downto 0);
    signal dec_rdy: std_logic;
    signal enc_rdy: std_logic;

  U1: rc5_key PORT MAP(clr=>clr, clk=>clk, key_vld=>key_rdy, key=>key, skey=>skey, key_rdy=>key_rdy);
  U2: rc5_enc PORT MAP(clr=>clr, clk=>clk, di_vld=>key_rdy, din=>din, skey=>skey, dout=>dout_enc,
                       do_rdy=>enc_rdy);
  U3: rc5_dec PORT MAP(clr=>clr, clk=>clk, di_vld=>key_rdy, din=>din, skey=>skey,  dout=>dout_dec,
                       do_rdy=>dec_rdy);

WITH enc SELECT
  dout<=dout_enc WHEN '1',
        dout_dec WHEN OTHERS;
WITH enc SELECT
  data_rdy<=enc_rdy WHEN '1',
            dec_rdy WHEN OTHERS;
END struct;
```

# Instantiate Components - Explained

- Three components are declared, rc5_enc, rc5_dec, and rc5_key and are exactly the same as the entities used for their respective designs.
- The architecture declares these three components and connects them together using signals. The syntax used is to first identify the component that is being instantiated using a keyword, in this case "u1", followed by the entity name for the component being instantiated. Then the keyword "port map" is used, followed by a list of signals to connect to the entity of the component. The signals in the design are connected to the components entity by the "=>" symbol. The name in front of the "=>" is the entity of the component and the name inback of the "=>" is what this port is connected to.
- For Example, When we are connecting the RC5 encryption "do_rdy" port to the top level enc_rdy, we use the following syntax in to component instantiation: do_rdy=>enc_rdy.
- We use the enc signal to define which set of outputs to use, that is, if we are performing encryption or decryption.
- NOTE: The design performs both encryption and decryption in parallel with each other and just selects the desired output using the enc signal to select between them.

# A Complete RC5 Module

- It works, but we can make it better
  - Encryption and decryption modules are similar, could we combine them together?
  - The maximum frequency is too low, could we do any better?
  - We must wait for 12 rounds – low throughput! How to improve it?
- Don't worry, you are not required to finish these three tasks for now



# Exercise

- Finish RC5 module
- Simulate both encryption and decryption
  - Functional Simulation using ModelSim
  - Code debugging
- Synthesize the Design
  - Perform Timing Simulation
- Understand following concepts/operations
  - Component declaration and instantiation
  - How to control the data flows between the modules?