C.N.E

ملخص

«مادة الميد»

Architecture & Organization Lab.

www.facebook.com/groups/CNE.FET

www.cnefamily.com

#TOGETHER_WE_REACH

CNEFAMILY

#اللجنة_الأكاديمية

* بعض الكلمات المحجوزة

- C <= A and B
  - جملة تعيين
- C <= A or B
- C <= Not A
- C <= A xor B
- C <= A xNor B
- C <= A NAND B
- C <= A nor B

* Variables Types:

- In
- Out

* اذا كان المتغير من bit واحد يتم تعريفه ←

اسم المتغير

A : in std_logic;

نوعه

* اذا كان المتغير أكثر من bit ← 3 bits

A : in std_logic_vector(2 downto 0);

عدد الـ bits

Entity Declaration

Syntax:

```
Entity name_of_entity is
port ( Declaring variable );

End name_of_entity;
```


Scanned by CamScanner

Syntax: Architecture body

```
Architecture arch-name of entity-name is
Begin
  ⋮
End arch-name;
```

Signals

Ex: متغيرات يتم استخدامها في البرنامج للتسهيل

```
Signal x, y : std-logic;
```

يتم تعريفها بال architecture قبل ال Begin

Ex: write a code to do: $F = A \oplus B$

```
Library IEEE;
use IEEE.std_logic_1164.all;
Entity xor-gate is
port(A, B : in std-logic;
     F : out std-logic);
End xor-gate;
Architecture prog of xor-gate is
Begin
F<= A xor B;
F<= (NOT A and B) or (A and NOT B)
End prog;
```

ثوابت بأي برنامج لاستدعاء المكتبة اللازمة

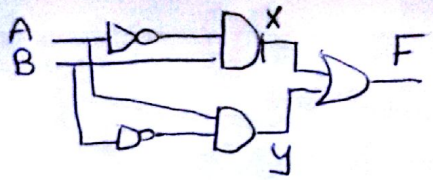
لا نضع ;

طريقة أخرى

* يتم وضع process اذا كان عندي شرط أو Loop

Scanned by CamScanner

ÿ Use signal to write code to do:




```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity XOR1 is
Port (A, B : in std_logic;
        F : out std_logic);
End XOR1

Architecture prog1 of XOR1 is
Signal x, y : std_logic;
Begin
x <= (Not A) and B;
y <= (Not B) and A;
F <= x or y;
End prog1;
```

ÿ EX D flip flop ⇒ In = Out when clock = 1

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity DFF1 is
port (data_in, clock : in std_logic;
      data_out : out std_logic);
End DFF1;
Architecture prog of DFF1 is
Begin
process (data in, clock)
 Begin
if (clock = '1' and clock'event) then
   data_out <= data_in;
End if;
End process;
End prog;
```

inputs

G.N.E computer & networks engineers

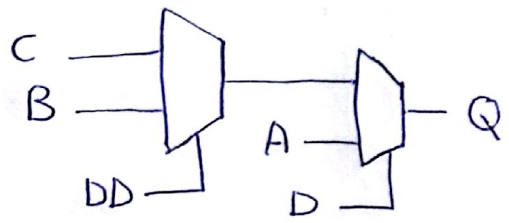
Conditional signal assignments

Syntax:

```
Signal_name <= Value when condition else
               Value2 when condition else
                   :
               Value ;
```

Ex:

```
Q <= A when D = '1' else
     B when DD = '1' else
     C;
```



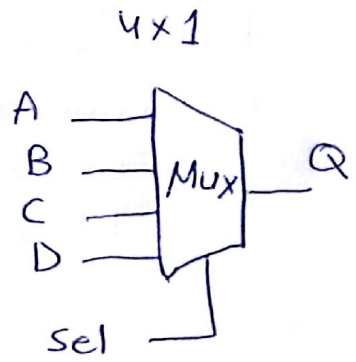
Selected signal assignments

syntax:

```
with expression select
signal_name <= value1 when condition1 ;
               value2 when condition2 ;
                   :
               value when others;
```

Ex:

```
whith sel select
Q <= A when "00";
     B when "01";
     C when "10";
     D when others;
```



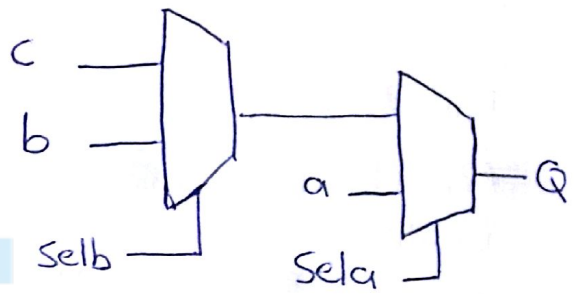
C.N.E
computer & network engineers

Scanned by CamScanner

IF - then statements

Syntax :

```
IF condition1 then
        :
ELSIF condition2 then
        :
Else
        :
End if ;
```



Ex :

```
IF sela = '1' then
   Q <= a ;
ELSIF selb = '1' then
   Q <= b ;
Else
   Q <= c ;
End if ;
```

وتوضع داخل process

C.N.E
computer & networks engineers

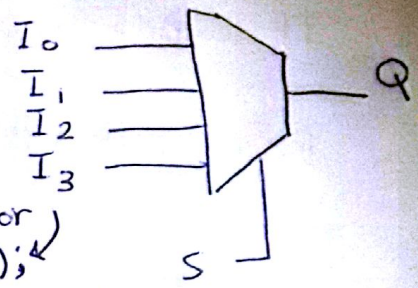
Case statement

syntax :

```
Case expression is
  when condition 1 =>
        :
  when condition 2 =>
        :
  when others =>
        :
  End case ;
```


Scanned by CamScanner

Ex: Design 4x1 mux in VHDL code using 3 bit vector I/O and case statement.




```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity Mux1 is
port ( I0, I1, I2, I3 : in std_logic_vector
                        (2 downto 0);
       s : in std_logic_vector(1 downto 0);
       Q : out std_logic_vector(2 downto 0));
End Mux1;
Architecture prog of Mux1 is
Begin
process( I0, I1, I2, I3 , S)
 Begin
case S is
when "00" => Q <= I0;
when "01" => Q <= I1;
when "10" => Q <= I2;
 when other => Q <= I3;
End case;
End process;
End prog;
```

C.N.E
computer & networks engineers

Scanned by CamScanner

Sructual description

- عملية استدعاء برنامج آخر
- Syntax and Ex

```
architecture Structual of xorGate is
  Component AND2 ---> 1 output , 2 Input برنامج آخر فيه
    port (in1 , in2 : in std_logic;
          out1 : out std_logic);
  End component;
Begin
U0: AND2 portmap (A , B , F);
End strctual;
```

Library and packages

- std_logic_1164 Package : Standard datatyper
- std_logic_arith package : arithmetic, conversion and comparison.
- std_logic_unsigned package
- std_logic_signed package

* To use any of these :

```
library IEEE;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_arith.all;
Use IEEE.std_logic_unsigned.all;
```

C.N.E computer & networks engineers

Scanned by CamScanner

State machine (FSM)

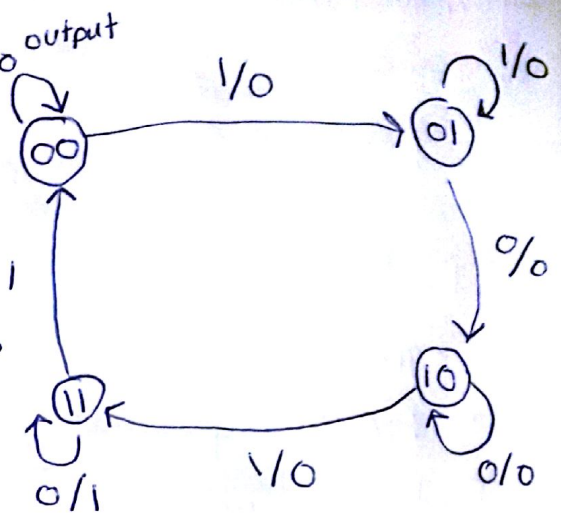
* يجب تحديد الـ Input و output ومتى رح يشتغل
* " " " الـ next state

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity FSM1 is
Port (a : in std_logic;
      clock : in std_logic;
      reset : in std_logic;
      x : out std_logic);
End FSM1;
```

input

رح تشتغل متى نتحرك

دائماً 1




```
Architecture Fsm2 of FSM1 is
Signal next_state, current : std_logic_vector(1 downto 0);
 Begin
 process (clock, reset)
  Begin
  if (reset='1') then
  current <= "00";
  elsif (clock='1' and clock'event) then
    current <= next_state;
  End if;
  End process;
 process(current, a)
 Begin
  case current is
  when "00" => x <= '0';
   if a='0' then
    next_state <= "00";
    elsif a='1' then
    next_state <= "01";
    end if;
```

G.N.E
computer & networks engineers

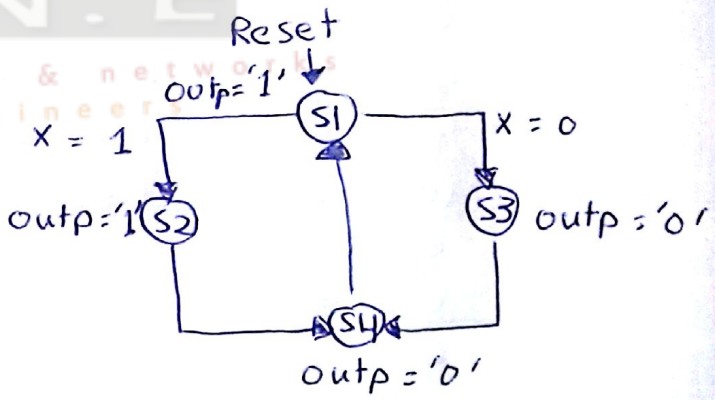
Scanned by CamScanner

```
when "01" => x<='0';
if a='0' then
next_state <= "10";
elsif a='1' then
next_state <= "01";
end if;

when "10" => x<='0';
if a='0' then
next_state <= "10";
elsif a='1' then
next_state <= "11";
end if;
when "11" => x<="1";
if a='0' then
next_state <= "11";
elsif a='1' then
next_state <= "00";
end if;
end case;
end process;
end Fsm2;
```

C.N.E
computer & networks engineer

Ex:



1. write a VHDL code with single process
2. with two process
3. with three process

Scanned by CamScanner

Single process

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity fsm is
port ( clk, reset, X: in std_logic;
       outp: out std_logic);
End fsm;
architecture fsm1 of fsm is
type state_type is (s1, s2, s3, s4);
Signal state: state_type ;
Begin
process (clk, reset)
Begin
if (reset = '1') then
state <= s1;
outp <= '1';
elsif (clk = '1' and clk'event) then
Case state is
when s1 => if x = '1' then state <= s2;
else state <= s3;
end if
outp <= '1';
when s2 => state <= s4; outp <= '1';
when s3 => state <= s4; outp <= '0';
when s4 => state <= s1 ; outp <= '0';
end case;
end if;
end process;
end fsm1;
```


Scanned by CamScanner

two process

③ نفس ال entity و signal

```
architecture fsm1 of FSM is
 process1 : process(clk,reset)
 Begin
 if(reset='1') then state<=s1;
 elsif(clk='1' and clk'event) then
  case state is
  when s1 => if x='1' then state=s2;
   else state <= s3;
   end if
   when s2 => state <= s4;
   when s3 => state <= s4;
   when s4 => state <= s1;
   end case;
   end if;
   end process process1;

  process2 : process(state)
   Begin
   case state is
   when s1 => outp<='1';
   when s2 => outp<='1';
   when s3 => outp<='0';
   when s4 => outp<='0';
   end case;
   end process process2;
   end fsm1;
```

C.N.E
computer & networks engineers

Scanned by CamScanner

three process

```
process1 : process( clk, reset)
 Begin
 if (reset = '1') then
  state <= s1;
elsif ( clk = '1' and clk 'event) then
 state <= next_state;
 End if;
 End process process1;
process2 : process (state, x)
 Begin
 case state is
  when s1 => if x = '1' then
        next_state <= s2;
   else next_state <= s3;
   end if;
  when s2 => next_state <= s4;
  when s3 => next_state <= s4;
  when s4 => next_state <= s1;
 End case;
 End process process2;
process3 : process (state)
 Begin
 Case state is
 when s1 => outp <= '1';
 when s2 => outp <= '1';
 when s3 => outp <= '0';
 when s4 => outp <= '0';
  End case;
  End process process3;
  End Fsm1 ;
```

* نفس ال Entity و
لكن زيادة Signal
signal next_state

CNE
computer & networks engineers

Scanned by CamScanner