

## Practical VHDL samples

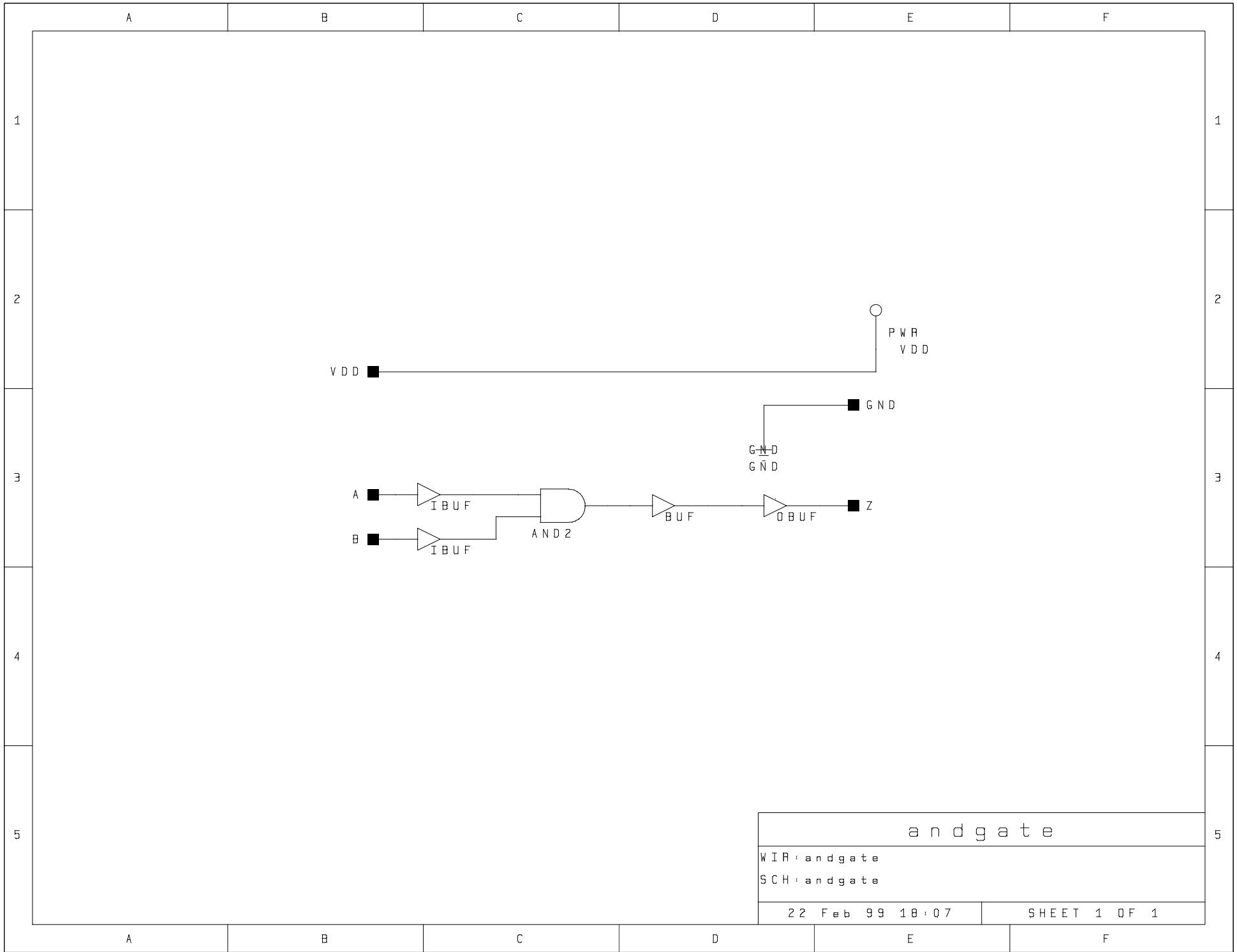
The following is a list of files used as examples in the ESD3 lectures.

The files are included overleaf with simulations and also post-synthesis schematics.

The target synthesis library is the Xilinx 4000 series of FPGA's- details of all the components are given at the end.

Source Name	Entity Name	Description	Synthesisable?	Comments
andgate.vhd	andgate	Simple AND Gate	✓	Implements a simple AND gate. Illustrates a very simple VHDL source code file- with entity and architecture.
pencoder.vhd	pencoder	2 Input Priority Encoder	✓	Implements a simple 2 input priority encoder. Illustrates the use of IF-THEN-ELSE as a prioritised selector.
mux.vhd	mux	2->1 Multiplexer	✓	Implements a simple 2->1 multiplexer with selection input. Demonstrates the use of the CASE statement as an equal priority selector.
simpreg.vhd	simpreg	Simple 8 Bit Register	✓	Implements a simple 8-bit register. Illustrates the inference of loadable registers etc.
par2ser.vhd	par2ser	Parallel to Serial Converter	✓	Implements a simple parallel-serial converter- with load and shift left modes. Illustrates the use of the FOR loop to facilitate multiple access operations. Also illustrates the use of internal signals.

```
-----  
-- Title       : Simple AND Gate Instantiation in VHDL  
-- Project     : Digital Design IV  
-----  
-- File        : andgate.vhd  
-- Author      : <Craig Slorach@HANDEL>  
-- Created     : 1999/02/05  
-- Last modified : 1999/02/05  
-----  
-- Description :  
-- Implements a simple AND gate in VHDL- used to highlight both entity  
-- and internal architecture.  
-----  
-- Modification history :  
-- 1999/02/05 : created  
-----  
  
library ieee;  
use ieee.std_logic_1164.all;  
  
entity ANDGATE is  
    port (A,B : in std_logic;          -- data inputs  
          Z : out std_logic);         -- data output  
  
end ANDGATE;  
  
-- purpose: Implement architecture  
architecture MYARCH of ANDGATE is  
  
begin -- MYARCH  
    Z <= A and B;  
  
end MYARCH;
```



andgate	
WIR: andgate	
SCH: andgate	
22 Feb 99 18:07	SHEET 1 OF 1

```
-----  
-- Title       : Priority encoder (illustrate IF-THEN-ELSE)  
-- Project     : Digital Design IV  
-----  
-- File       : pencoder.vhd  
-- Author     : <craigs@HANDEL>  
-- Created    : 1999/02/19  
-- Last modified : 1999/02/19  
-----  
-- Description :  
-- Implements a simple priority encoder in VHDL. This code demonstrates  
-- a simple examples of the IF-THEN-ELSE statement as a prioritised  
-- selector.  
-----  
-- Modification history :  
-- 1999/02/19 : created  
-----
```

```
library ieee;  
use ieee.std_logic_1164.all;
```

```
entity PENCODER is
```

```
    port (DATAIN : in std_logic_vector(1 downto 0); -- data input  
          DATAOUT : out std_logic; -- data out  
          CLK,RESET : in std_logic); -- clock and reset
```

```
end PENCODER;
```

```
-- purpose: Implement main architecture of PENCODER  
architecture BEHAVIOR of PENCODER is
```

```
begin -- BEHAVIOR
```

```
    -- purpose: Main process  
    process (CLK, RESET)
```

```
    begin -- process
```

```
        -- activities triggered by asynchronous reset (active high)  
        if RESET = '1' then
```

```
            DATAOUT <= '0'; -- default output
```

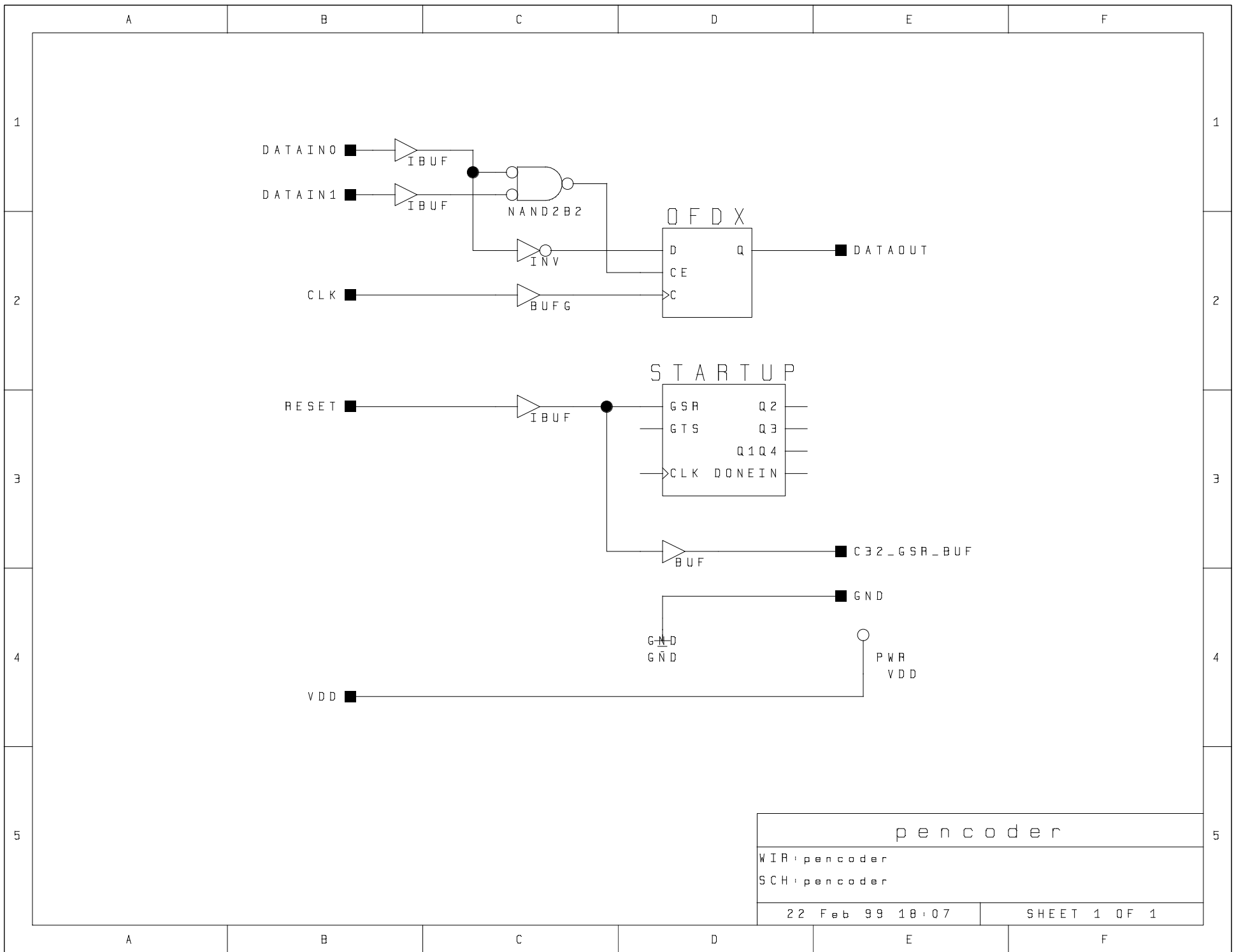
```
        -- activities triggered by rising edge of clock  
        elsif CLK'event and CLK = '1' then
```

```
            if DATAIN(0)='1' then  
                DATAOUT <= '0';  
            elsif DATAIN(1)='1' then  
                DATAOUT <= '1';  
            end if;
```

```
        end if;
```

```
    end process;
```

```
end BEHAVIOR;
```



```
-----
-- Title       : Multiplexer in VHDL
-- Project     : Digital Design IV
-----
-- File        : mux.vhd
-- Author      : <craigs@HANDEL>
-- Created     : 1999/02/19
-- Last modified : 1999/02/19
-----
-- Description :
-- Implement a simple 2->1 multiplexer in VHDL.
-----
-- Modification history :
-- 1999/02/19 : created
-----

library ieee;
use ieee.std_logic_1164.all;

entity MUX is
    port (SEL : in std_logic;    -- select input
          DATAIN : in std_logic_vector(1 downto 0); -- Input data
          DATAOUT : out std_logic); -- Output data
end MUX;

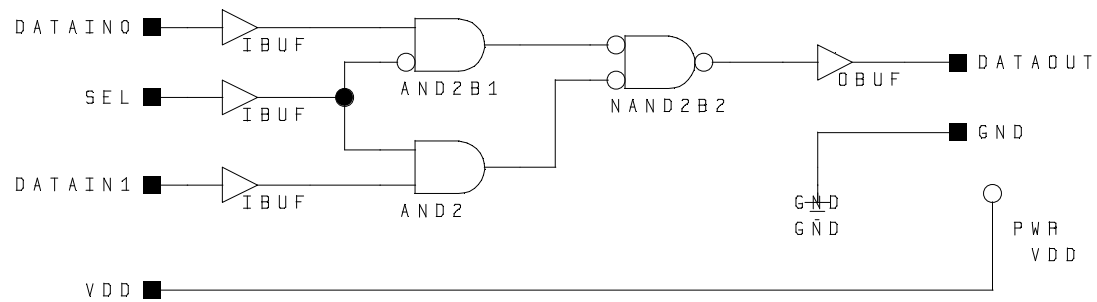
-- purpose: Implement main architecture of MUX
architecture BEHAVIOR of MUX is
begin -- BEHAVIOR

    -- purpose: Main process
    -- type: memoryless
    -- inputs:
    -- outputs:
    process (SEL,DATAIN)

    begin -- process

        case SEL is
            when '0' =>
                DATAOUT <= DATAIN(0);
            when others =>
                DATAOUT <= DATAIN(1);
        end case;

    end process;
end BEHAVIOR;
```



M U X	
WIR: mux	
SCH: mux	
22 Feb 99 18:07	SHEET 1 OF 1

```
-----
-- Title       : Simple Register Example
-- Project     : Digital Design IV
-----
-- File        : simpreg.vhd
-- Author      : <Craig Slorach@HANDEL>
-- Created     : 1999/02/02
-- Last modified : 1999/02/02
-----
-- Description :
-- Implements a simple loadable register in VHDL
-----
-- Modification history :
-- 1999/02/02 : created
-----

library ieee;
use ieee.std_logic_1164.all;

entity SIMPREG is

    port (DIN : in std_logic_vector(7 downto 0); -- system inputs
          DOUT : out std_logic_vector(7 downto 0); -- system outputs
          ENABLE : in std_logic; -- enable
          CLK,RESET : in std_logic); -- clock and reset

end SIMPREG;

-- purpose: Main architecture details for SIMPREG
architecture SIMPLE of SIMPREG is

begin -- SIMPLE

    process(CLK,RESET)

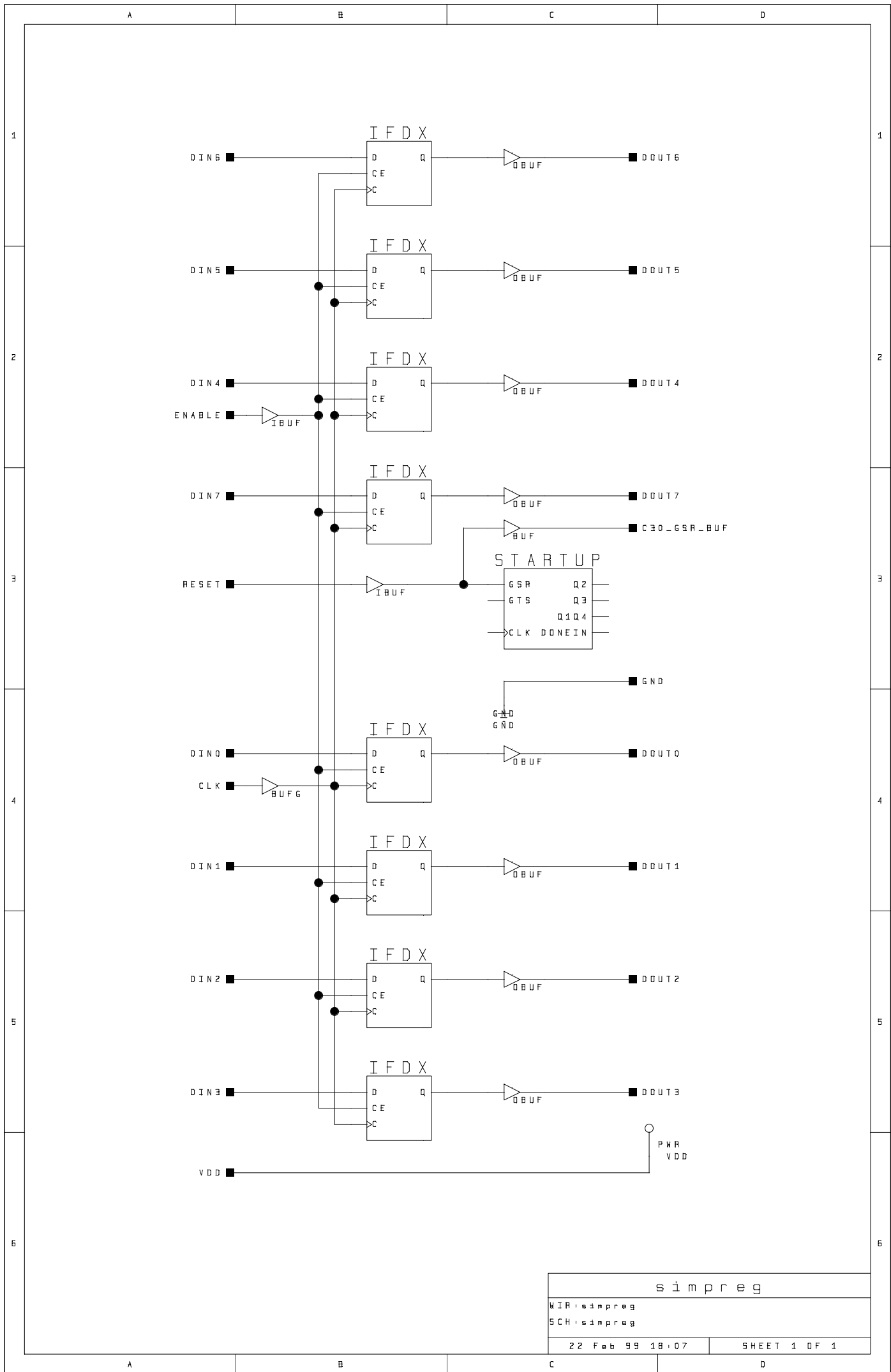
    begin -- process
        -- activities triggered by asynchronous reset (active high)
        if RESET = '1' then
            DOUT <= "00000000";

            -- activities triggered by rising edge of clock
            elsif CLK'event and CLK = '1' then

                if ENABLE='1' then
                    DOUT <= DIN;
                else
                    null;
                end if;

            end if;
        end process;

    end SIMPLE;
```



simplereg	
WIR:simplereg	
SCH:simplereg	
22 Feb 99 18:07	SHEET 1 OF 1

```
-----
-- Title       : Parallel to Serial Converter (PAR2SER)
-- Project     : Digital Design IV
-----
-- File        : par2ser.vhd
-- Author      : <craigs@HANDEL>
-- Created     : 1999/02/19
-- Last modified : 1999/02/19
-----
-- Description :
-- Implements a simple 8-bit parallel to serial converter in VHDL.
-----
-- Modification history :
-- 1999/02/19 : created
-----

library ieee;
use ieee.std_logic_1164.all;

entity PAR2SER is

    port (DIN : in std_logic_vector (7 downto 0);           -- input register
          MODE : in std_logic_vector (1 downto 0);         -- mode selection
          CLK, RESET : in std_logic;                       -- clock and reset
          SDOUT : out std_logic);                          -- output data

end PAR2SER;

-- purpose: Implement main architecture of PAR2SER
architecture BEHAVIOR of PAR2SER is

    signal IDATA : std_logic_vector(7 downto 0); -- internal data

begin -- BEHAVIOR

    -- purpose: Main process
    process (CLK, RESET)

        begin -- process

            -- activities triggered by asynchronous reset (active high)
            if RESET = '1' then

                SDOUT <= '0';
                IDATA <= "00000000";

            -- activities triggered by rising edge of clock
            elsif CLK'event and CLK = '1' then

                case MODE is

                    when "00" => -- no operation
                        null;

                    when "01" => -- load operation
                        IDATA <= DIN;

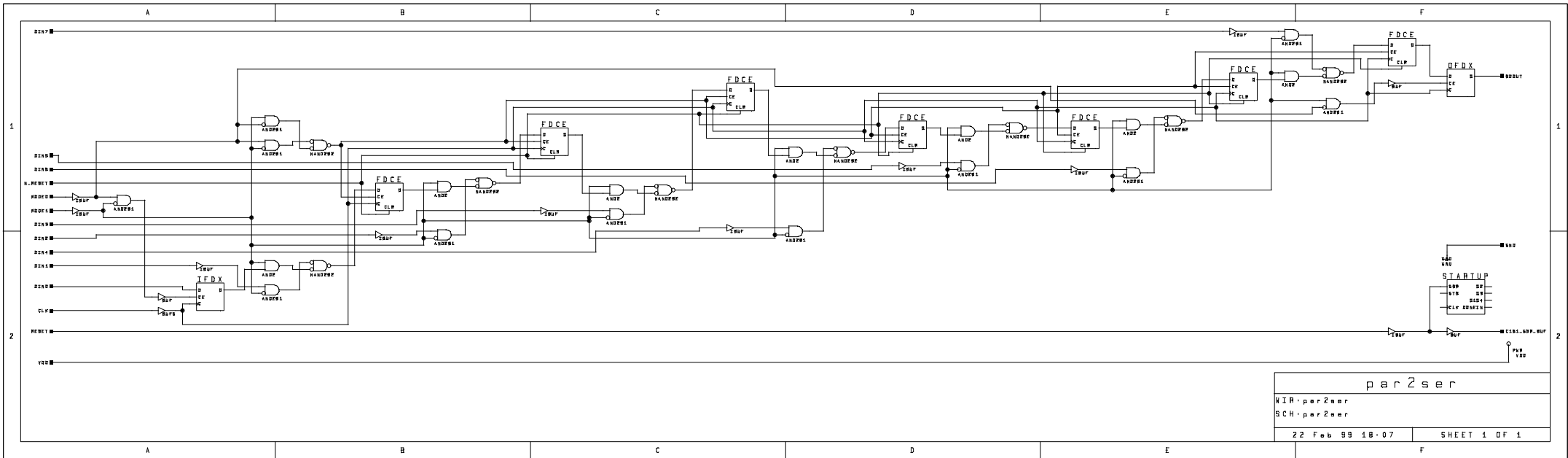
                    when "10" => -- shift left
                        SDOUT <= IDATA(7);

                        for mloop in 6 downto 0 loop
                            IDATA(mloop+1) <= IDATA(mloop);
                        end loop; -- mloop

                    when others => -- no operation otherwise
                        null;

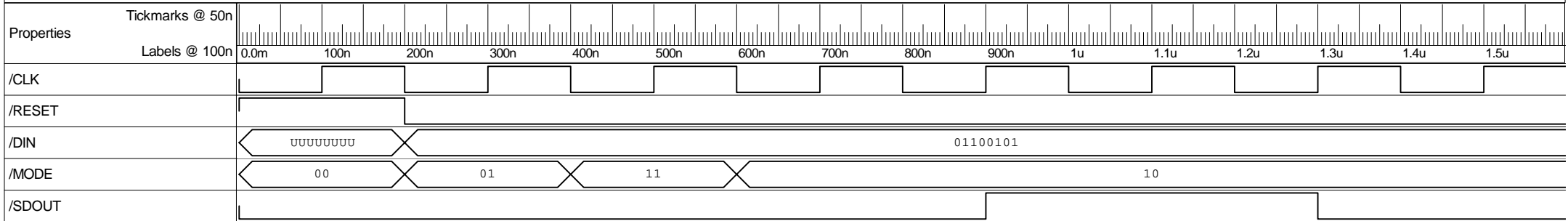
                end case;
            end if;
        end process;

end BEHAVIOR;
```



par2ser	
VIR.par2ser	
SCH.par2ser	
22 Feb 99 18:07	SHEET 1 OF 1

Plot of Parallel to Serial Example (PAR2SER.VHD)



## IBUF, 4, 8, 16

### Single- and Multiple-Input Buffers

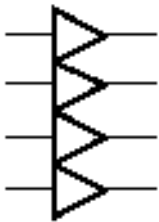
Element	XC3000	XC4000 E	XC4000 X	XC5200	XC9000
IBUF	Primitive	Primitive	Primitive	Primitive	Primitive
IBUF4, IBUF8, IBUF16	Macro	Macro	Macro	Macro	Macro

IBUF



X3784

IBUF4



X3791

IBUF8



X3803

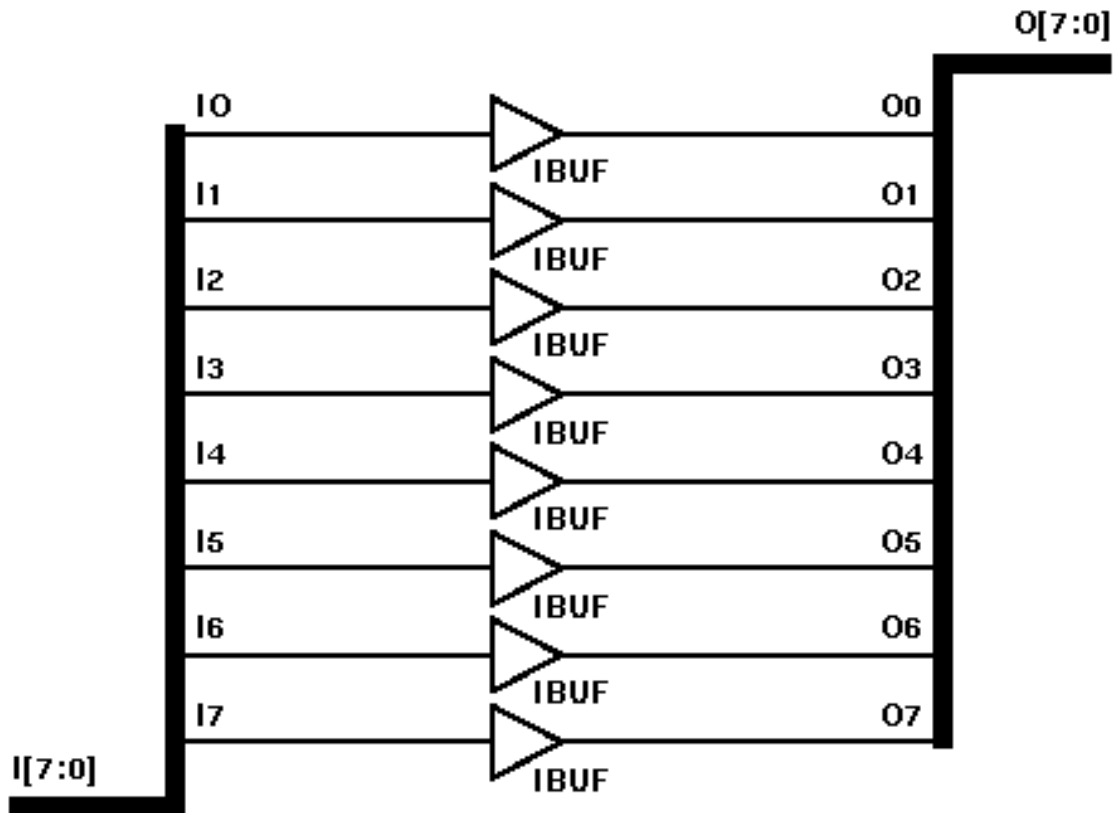
IBUF16



X3815

IBUF, IBUF4, IBUF8, and IBUF16 are single and multiple input buffers. An IBUF isolates the internal circuit from the signals coming into a chip. IBUFs are contained in input/output blocks (IOB). IBUF inputs (I) are connected to an IPAD or an IOPAD. IBUF outputs (O) are connected to the internal circuit.

Figure 6-3IBUF8 Implementation XC3000, XC4000, XC5200, XC9000

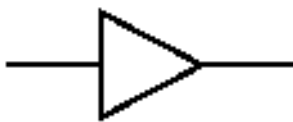


X7652

## BUF

### General-Purpose Buffer

XC3000	XC4000E	XC4000X	XC5200	XC9000
Primitive	Primitive	Primitive	Primitive	Primitive

**X3830**

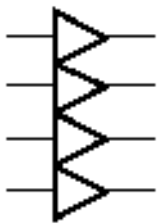
BUF is a general purpose, non-inverting buffer.

In FPGA architecture, BUF is usually not necessary and is removed by the partitioning software (MAP for XC3000 and PAR for XC4000). The BUF element can be preserved for reducing the delay on a high fan-out net, for example, by splitting the net and reducing capacitive loading. In this case, the buffer is preserved by attaching an X (explicit) attribute to both the input and output nets of the BUF.

In CPLD architecture, BUF is usually removed, unless you inhibit optimization by applying the OPT=OFF attribute to the BUF symbol or by using the LOGIC\_OPT=OFF global attribute.

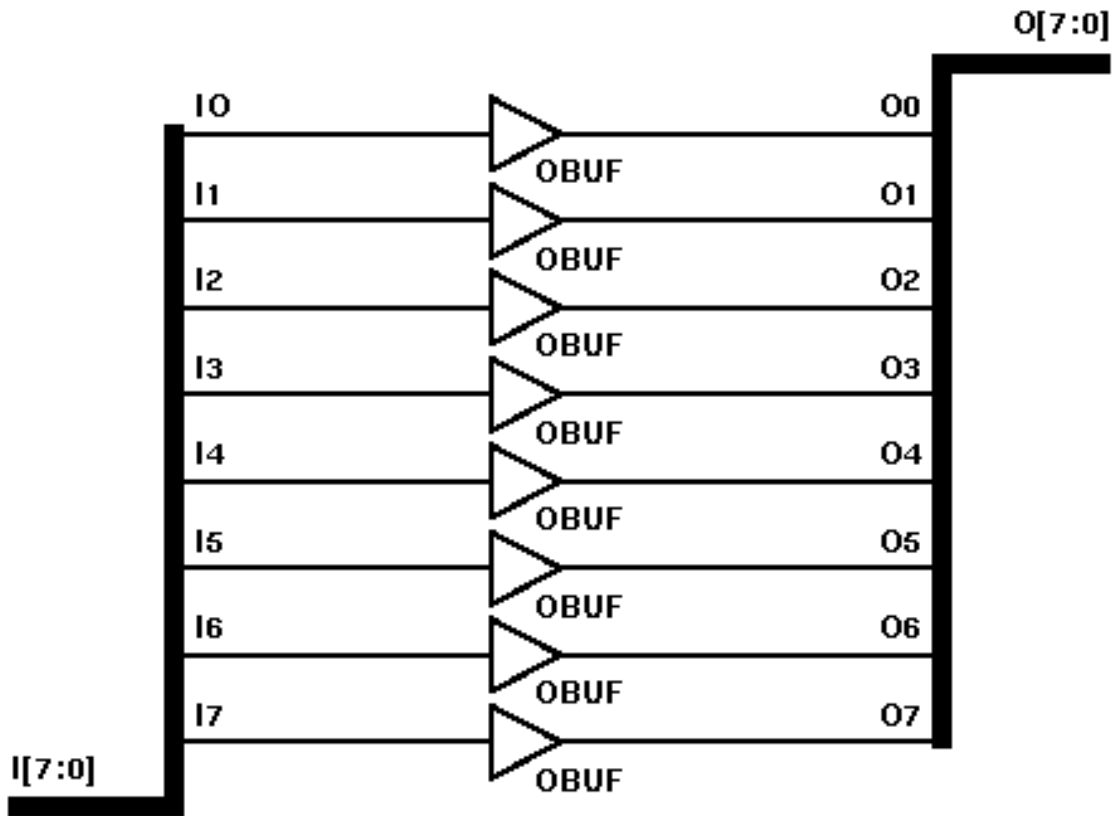
**OBUF, 4, 8, 16****Single- and Multiple-Output Buffers**

Element	XC3000	XC4000 E	XC4000 X	XC5200	XC9000
OBUF	Primitive	Primitive	Primitive	Primitive	Primitive
OBUF4, OBUF8, OBUF16	Macro	Macro	Macro	Macro	Macro

**OBUF****X3785****OBUF4****X3792****OBUF8****X3804****OBUF16****X3816**

OBUF, OBUF4, OBUF8, and OBUF16 are single and multiple output buffers. An OBUF isolates the internal circuit and provides drive current for signals leaving a chip. OBUFs exist in input/output blocks (IOB). The output (O) of an OBUF is connected to an OPAD or an IOPAD. For XC9000 CPLDs, if a high impedance (Z) signal from an on-chip 3-state buffer (like BUFE) is applied to the input of an OBUF, it is propagated to the CPLD device output pin.

**Figure 8-1 OBUF8 Implementation XC3000, XC4000, XC5200, XC9000**

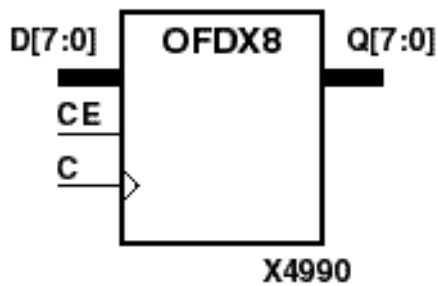
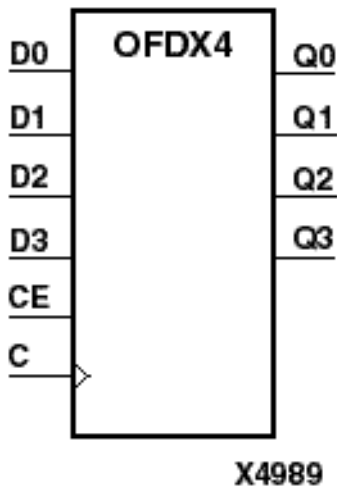
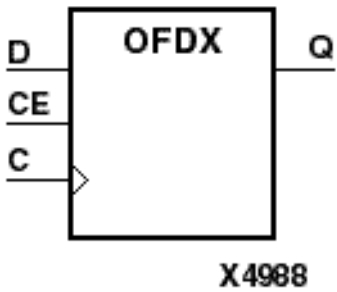


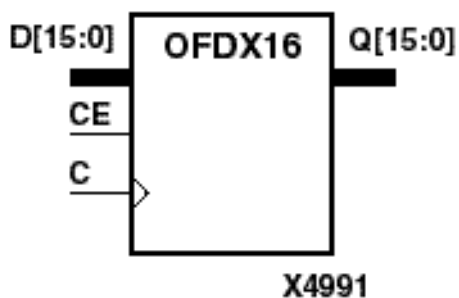
X7654

# OFDX, 4, 8, 16

## Single- and Multiple-Output D Flip-Flops with Clock Enable

Element	XC3000	XC4000 E	XC4000 X	XC5200	XC9000
OFDX	N/A	Primitive	Primitive	N/A	N/A
OFDX4, OFDX8, OFDX16	N/A	Macro	Macro	N/A	N/A





OFDX, OFDX4, OFDX8, and OFDX16 are single and multiple output D flip-flops. The flip-flops are located in an input/output block (IOB) for XC4000E. The Q outputs are connected to OPADs or IOPADs. The data on the D inputs is loaded into the flip-flops during the Low-to-High clock (C) transition and appears on the Q outputs. When CE is Low, flip-flop outputs do not change.

The flip-flops are asynchronously cleared with Low outputs, when power is applied or when global set/reset (GSR) is active. The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP symbol.

Inputs			Outputs
CE	D	C	Q
1	D	↑	dn
0	X	X	No Chg

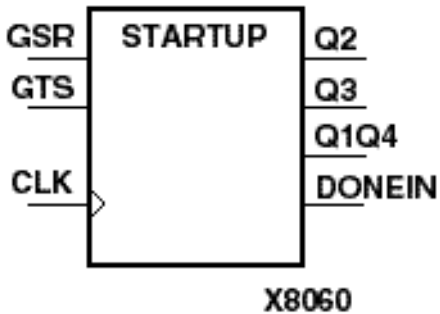
dn = state of referenced input one setup time prior to active clock transition

**Figure 8-35** OFDX8 Implementation XC4000

## STARTUP

### User Interface to Global Clock, Reset, and 3-State Controls

XC3000	XC4000E	XC4000X	XC5200	XC9000
N/A	Primitive	Primitive	Primitive	N/A



The STARTUP primitive is used for Global Set/Reset, global 3-state control, and the user configuration clock. The Global Set/Reset (GSR) input, when High, sets or resets every flip-flop in the device, depending on the initialization state (S or R) of the flip-flop. Following configuration, the global 3-state control (GTS), when High, forces all the IOB outputs into high impedance mode, which isolates the device outputs from the circuit but leaves the inputs active.

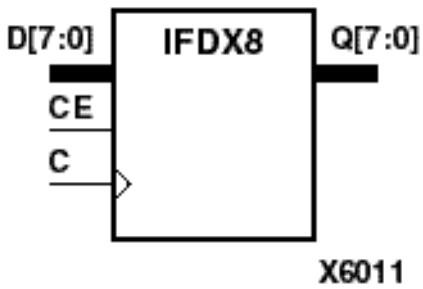
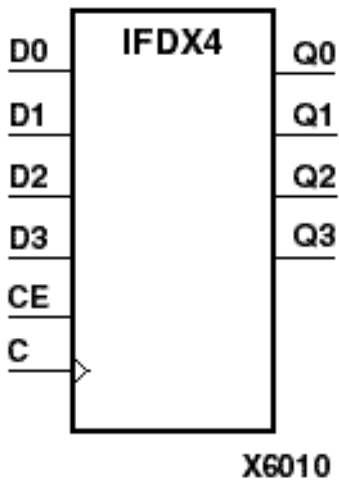
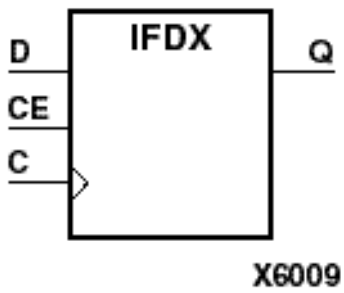
Including the STARTUP symbol in a design is optional. You must include the symbol under the following conditions.

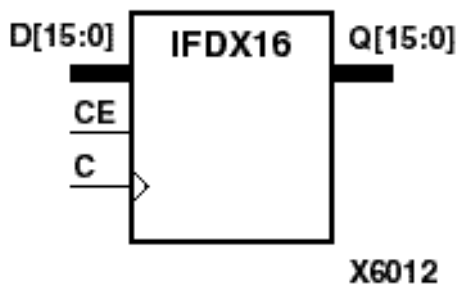
- If you intend to exert external control over global set/reset, you must connect the GSR pin to an IPAD and an IBUF, as shown here.

# IFDX, 4, 8, 16

## Single- and Multiple-Input D Flip-Flops with Clock Enable

Element	XC3000	XC4000 E	XC4000 X	XC5200	XC9000
IFDX	N/A	Primitive	Primitive	N/A	N/A
IFDX4, IFDX8, IFDX16	N/A	Macro	Macro	N/A	N/A





The IFDX D-type flip-flop is contained in an input/output block (IOB). The input (D) of the flip-flop is connected to an IPAD or an IOPAD (without using an IBUF). The D input provides data input for the flip-flop, which synchronizes data entering the chip. The data on input D is loaded into the flip-flop during the Low-to-High clock (C) transition and appears at the output (Q). The clock input can be driven by internal logic or through another external pin. When CE is Low, flip-flop outputs do not change.

The flip-flops are asynchronously cleared with Low outputs when power is applied or when global set/reset (GSR) is active. The GSR active level defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP symbol.

For information on legal IFDX, IFDX\_1, ILDX, and ILDX\_1 combinations, refer to the "**ILDX, 4, 8, 16**" section.

Inputs			Outputs
CE	Dn	C	Qn
1	Dn	↑	dn
0	X	X	No Chg

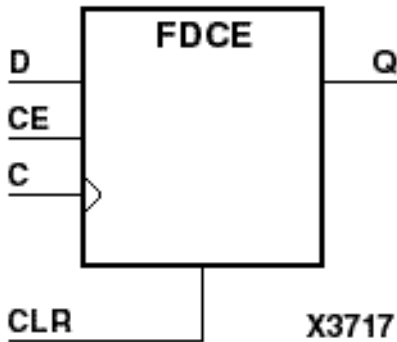
dn = state of referenced input (Dn) one setup time prior to active clock transition

**Figure 6-12IFDX8 Implementation XC4000**

## FDCE

### D Flip-Flop with Clock Enable and Asynchronous Clear

XC3000	XC4000E	XC4000X	XC5200	XC9000
Primitive	Primitive	Primitive	Primitive	Macro



FDCE is a single D-type flip-flop with clock enable and asynchronous clear. When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data input (D) of FDCE is transferred to the corresponding data output (Q) during the Low-to-High clock (C) transition. When CLR is High, it overrides all other inputs and resets the data output (Q) Low. When CE is Low, clock transitions are ignored.

The flip-flop is asynchronously cleared, output Low, when power is applied or when global reset (GR for XC3000, XC5200), global set/reset (GSR for XC4000), or global set/reset preload (PRLD for XC9000) is active. GR for XC3000 is active-Low. PRLD is active-High. The active level of the GSR and of the GR for XC5200 defaults to active-High but can be inverted by adding an inverter in front of the GSR/GR input of the STARTUP symbol.

Inputs				Outputs
CLR	CE	D	C	Q
1	X	X	X	0
0	0	X	X	No Chg
0	1	1	↑	1
0	1	0	↑	0

Figure 5-13 FDCE Implementation XC9000