Previous Page Next Page

Counters

XST is able to recognize counters with the following control signals.

- Asynchronous Set/Clear
- Synchronous Set/Clear
- Asynchronous/Synchronous Load (signal and/or constant)
- Clock Enable
- Modes (Up, Down, Up/Down)
- Mixture of all of the above

HDL coding styles for the following control signals are equivalent to the ones described in "Registers" in this chapter.

- Clock
- Asynchronous Set/Clear
- Synchronous Set/Clear
- Clock Enable

Moreover, XST supports both unsigned and signed counters.

Log File

The XST log file reports the type and size of recognized counters during the Macro Recognition step.

```
...
Synthesizing Unit <counter>.
     Related source file is counters_1.vhd.
     Found 4-bit up counter for signal <tmp>.
     Summary:
         inferred   1 Counter(s).
 Unit <counter> synthesized.
=============================
 HDL Synthesis Report
 Macro Statistics
 # Counters                       : 1
   4-bit  up  counter             : 1
=============================
...
```

Note: During synthesis, XST decomposes Counters on Adders and Registers if they do not contain synchronous load signals. This is done to create additional opportunities for timing optimization. Because of this, counters reported during the Macro Recognition step and in the overall statistics of recognized macros may not appear in the final report. Adders/registers are reported instead.

Related Constraints

There are no related constraints available.

4-bit Unsigned Up Counter with Asynchronous Clear

The following table shows pin definitions for a 4-bit unsigned up counter with an asynchronous clear.

IO Pins	Description
C	Positive-Edge Clock
CLR	Asynchronous Clear (active High)
Q[3:0]	Data Output

VHDL Code

Following is VHDL code for a 4-bit unsigned up counter with an asynchronous clear.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit unsigned up counter with an asynchronous clear.
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_1 is
    port(C, CLR : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_1;

architecture archi of counters_1 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C, CLR)
    begin
        if (CLR='1') then
            tmp <= "0000";
        elsif (C'event and C='1') then
            tmp <= tmp + 1;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned up counter with asynchronous clear.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit unsigned up counter with an asynchronous clear.
//

module v_counters_1 (C, CLR, Q);
```

```
    input C, CLR;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C or posedge CLR)
    begin
        if (CLR)
            tmp <= 4'b0000;
        else
            tmp <= tmp + 1'b1;
    end

    assign Q = tmp;
endmodule
```

.

4-bit Unsigned Down Counter with Synchronous Set

The following table shows pin definitions for a 4-bit unsigned down counter with a synchronous set.

IO Pins	Description
C	Positive-Edge Clock
S	Synchronous Set (active High)
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit unsigned down counter with a synchronous set.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit unsigned down counter with a synchronous set.
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_2 is
    port(C, S : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_2;

architecture archi of counters_2 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C)
    begin
        if (C'event and C='1') then
            if (S='1') then
                tmp <= "1111";
            else
                tmp <= tmp - 1;
            end if;
        end if;
```

```
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned down counter with synchronous set.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit unsigned down counter with a synchronous set.
//

module v_counters_2 (C, S, Q);
    input C, S;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C)
    begin
        if (S)
            tmp <= 4'b1111;
        else
            tmp <= tmp - 1'b1;
    end

    assign Q = tmp;

endmodule
```

.

4-bit Unsigned Up Counter with Asynchronous Load from Primary Input

The following table shows pin definitions for a 4-bit unsigned up counter with an asynchronous load from the primary input.

IO Pins	Description
C	Positive-Edge Clock
ALOAD	Asynchronous Load (active High)
D[3:0]	Data Input
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit unsigned up counter with an asynchronous load from the primary input.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Unsigned Up Counter with Asynchronous Load from Primary Input
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_3 is
    port(C, ALOAD : in std_logic;
         D : in std_logic_vector(3 downto 0);
         Q : out std_logic_vector(3 downto 0));
end counters_3;

architecture archi of counters_3 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C, ALOAD, D)
    begin
        if (ALOAD='1') then
            tmp <= D;
        elsif (C'event and C='1') then
            tmp <= tmp + 1;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned up counter with an asynchronous load from the primary input.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Unsigned Up Counter with Asynchronous Load from Primary Input
//

module v_counters_3 (C, ALOAD, D, Q);
    input C, ALOAD;
    input [3:0] D;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C or posedge ALOAD)
    begin
        if (ALOAD)
            tmp <= D;
        else
            tmp <= tmp + 1'b1;
    end

    assign Q = tmp;
```

```
endmodule
```

.

4-bit Unsigned Up Counter with Synchronous Load with a Constant

The following table shows pin definitions for a 4-bit unsigned up counter with a synchronous load with a constant.

IO Pins	Description
C	Positive-Edge Clock
SLOAD	Synchronous Load (active High)
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit unsigned up counter with a synchronous load with a constant.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Unsigned Up Counter with Synchronous Load with a Constant
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_4 is
    port(C, SLOAD : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_4;

architecture archi of counters_4 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C)
    begin
        if (C'event and C='1') then
            if (SLOAD='1') then
                tmp <= "1010";
            else
                tmp <= tmp + 1;
            end if;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned up counter with a synchronous load with a constant.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Unsigned Up Counter with Synchronous Load with a Constant
//

module v_counters_4 (C, SLOAD, Q);
    input C, SLOAD;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C)
    begin
        if (SLOAD)
            tmp <= 4'b1010;
        else
            tmp <= tmp + 1'b1;
    end

    assign Q = tmp;

endmodule
```

4-bit Unsigned Up Counter with Asynchronous Clear and Clock Enable

The following table shows pin definitions for a 4-bit unsigned up counter with an asynchronous clear and a clock enable.

IO Pins	Description
C	Positive-Edge Clock
CLR	Asynchronous Clear (active High)
CE	Clock Enable
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit unsigned up counter with an asynchronous clear and a clock enable.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Unsigned Up Counter with Asynchronous Clear and Clock Enable
--

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

```
use ieee.std_logic_unsigned.all;

entity counters_5 is
    port(C, CLR, CE : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_5;

architecture archi of counters_5 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C, CLR)
    begin
        if (CLR='1') then
            tmp <= "0000";
        elsif (C'event and C='1') then
            if (CE='1') then
                tmp <= tmp + 1;
            end if;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned up counter with an asynchronous clear and a clock enable.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Unsigned Up Counter with Asynchronous Clear and Clock Enable
//

module v_counters_5 (C, CLR, CE, Q);
    input C, CLR, CE;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C or posedge CLR)
    begin
        if (CLR)
            tmp <= 4'b0000;
        else if (CE)
            tmp <= tmp + 1'b1;
    end

    assign Q = tmp;

endmodule
```

.

4-bit Unsigned Up/Down counter with Asynchronous Clear

The following table shows pin definitions for a 4-bit unsigned up/down counter with an asynchronous clear.

IO Pins	Description
C	Positive-Edge Clock
CLR	Asynchronous Clear (active High)
UP_DOWN	up/down count mode selector
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit unsigned up/down counter with an asynchronous clear.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Unsigned Up/Down counter with Asynchronous Clear
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counters_6 is
    port(C, CLR, UP_DOWN : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_6;

architecture archi of counters_6 is
    signal tmp: std_logic_vector(3 downto 0);
begin
    process (C, CLR)
    begin
        if (CLR='1') then
            tmp <= "0000";
        elsif (C'event and C='1') then
            if (UP_DOWN='1') then
                tmp <= tmp + 1;
            else
                tmp <= tmp - 1;
            end if;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit unsigned up/down counter with an asynchronous clear.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Unsigned Up/Down counter with Asynchronous Clear
//

module v_counters_6 (C, CLR, UP_DOWN, Q);
    input C, CLR, UP_DOWN;
    output [3:0] Q;
    reg [3:0] tmp;

    always @(posedge C or posedge CLR)
    begin
        if (CLR)
            tmp <= 4'b0000;
        else if (UP_DOWN)
                tmp <= tmp + 1'b1;
            else
                tmp <= tmp - 1'b1;
    end

    assign Q = tmp;

endmodule
```

.

4-bit Signed Up Counter with Asynchronous Reset

The following table shows pin definitions for a 4-bit signed up counter with an asynchronous reset.

IO Pins	Description
C	Positive-Edge Clock
CLR	Asynchronous Clear (active High)
Q[3:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit signed up counter with an asynchronous reset.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Signed Up Counter with Asynchronous Reset
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;

entity counters_7 is
    port(C, CLR : in std_logic;
         Q : out std_logic_vector(3 downto 0));
end counters_7;

architecture archi of counters_7 is
    signal tmp: std_logic_vector(3 downto 0);
begin
```

```
    process (C, CLR)
    begin
        if (CLR='1') then
            tmp <= "0000";
        elsif (C'event and C='1') then
            tmp <= tmp + 1;
        end if;
    end process;

    Q <= tmp;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit signed up counter with an asynchronous reset.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Signed Up Counter with Asynchronous Reset
//

module v_counters_7 (C, CLR, Q);
    input C, CLR;
    output signed [3:0] Q;
    reg signed [3:0] tmp;

    always @ (posedge C or posedge CLR)
    begin
        if (CLR)
            tmp <= 4'b0000;
        else
            tmp <= tmp + 1'b1;
    end

    assign Q = tmp;

endmodule
```

.

4-bit Signed Up Counter with Asynchronous Reset and Modulo Maximum

The following table shows pin definitions for a 4-bit signed up counter with an asynchronous reset and a modulo maximum.

IO Pins	Description
C	Positive-Edge Clock
CLR	Asynchronous Clear (active High)
Q[7:0]	Data Output

VHDL Code

Following is the VHDL code for a 4-bit signed up counter with an asynchronous reset and a maximum using the VHDL mod function.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
--
-- 4-bit Signed Up Counter with Asynchronous Reset and Modulo Maximum
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity counters_8 is
    generic (MAX : integer := 16);
    port(C, CLR : in std_logic;
         Q : out integer range 0 to MAX-1);
end counters_8;

architecture archi of counters_8 is
    signal cnt : integer range 0 to MAX-1;
begin
    process (C, CLR)
    begin
        if (CLR='1') then
            cnt <= 0;
        elsif (rising_edge(C)) then
            cnt <= (cnt + 1) mod MAX ;
        end if;
    end process;

    Q <= cnt;

end archi;
```

.

Verilog Code

Following is the Verilog code for a 4-bit signed up counter with an asynchronous reset and a modulo maximum.

These coding examples are accurate as of the date of publication. You can download any updates to these examples from ftp://ftp.xilinx.com/pub/documentation/misc/examples_v7.zip

.

```
//
// 4-bit Signed Up Counter with Asynchronous Reset and Modulo Maximum
//

module v_counters_8 (C, CLR, Q);
    parameter
        MAX_SQRT = 4,
        MAX = (MAX_SQRT*MAX_SQRT);

    input   C, CLR;
    output [MAX_SQRT-1:0] Q;
    reg    [MAX_SQRT-1:0] cnt;

    always @ (posedge C or posedge CLR)
    begin
        if (CLR)
            cnt <= 0;
```

```
        else
            cnt <= (cnt + 1) %MAX;
    end

    assign Q = cnt;

endmodule
```

.

Related Constraints

There are no related constraints available.

www.xilinx.com