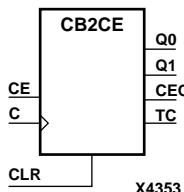


## CB2CE, CB4CE, CB8CE, CB16CE

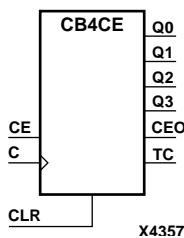
### 2-, 4-, 8-, 16-Bit Cascadable Binary Counters with Clock Enable and Asynchronous Clear

#### Architectures Supported

CB2CE, CB4CE, CB8CE, CB16CE	
Spartan-II, Spartan-IIIE	Macro
Spartan-3	Macro
Virtex, Virtex-E	Macro
Virtex-II, Virtex-II Pro, Virtex-II Pro X	Macro
XC9500, XC9500XV, XC9500XL	Macro
CoolRunner XPLA3	Macro
CoolRunner-II	Macro
CoolRunner-IIS	No



CB2CE, CB4CE, CB8CE, and CB16CE are, respectively, 2-, 4-, 8-, and 16-bit (stage), asynchronous, clearable, cascadable binary counters. The asynchronous clear (CLR) is the highest priority input. When CLR is High, all other inputs are ignored; the Q outputs, terminal count (TC), and clock enable out (CEO) go to logic level zero, independent of clock transitions. The Q outputs increment when the clock enable input (CE) is High during the Low-to-High clock (C) transition. The counter ignores clock transitions when CE is Low. The TC output is High when all Q outputs are High.



Larger counters are created by connecting the CEO output of the first stage to the CE input of the next stage and connecting the C and CLR inputs in parallel. CEO is active (High) when TC and CE are High. The maximum length of the counter is determined by the accumulated CE-to-TC propagation delays versus the clock period. The clock period must be greater than  $n(t_{CE-TC})$ , where  $n$  is the number of stages and the time  $t_{CE-TC}$  is the CE-to-TC propagation delay of each stage. When cascading counters, use the CEO output if the counter uses the CE input; use the TC output if it does not.

The counter is asynchronously cleared, outputs Low, when power is applied.

For XC9500/XV/XL, CoolRunner XPLA3, and CoolRunner-II, the power-on condition can be simulated by applying a High-level pulse on the PRLD global net.

Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP\_SPARTAN2, STARTUP\_SPARTAN3, STARTUP\_VIRTEX, or STARTUP\_VIRTEX2 symbol.

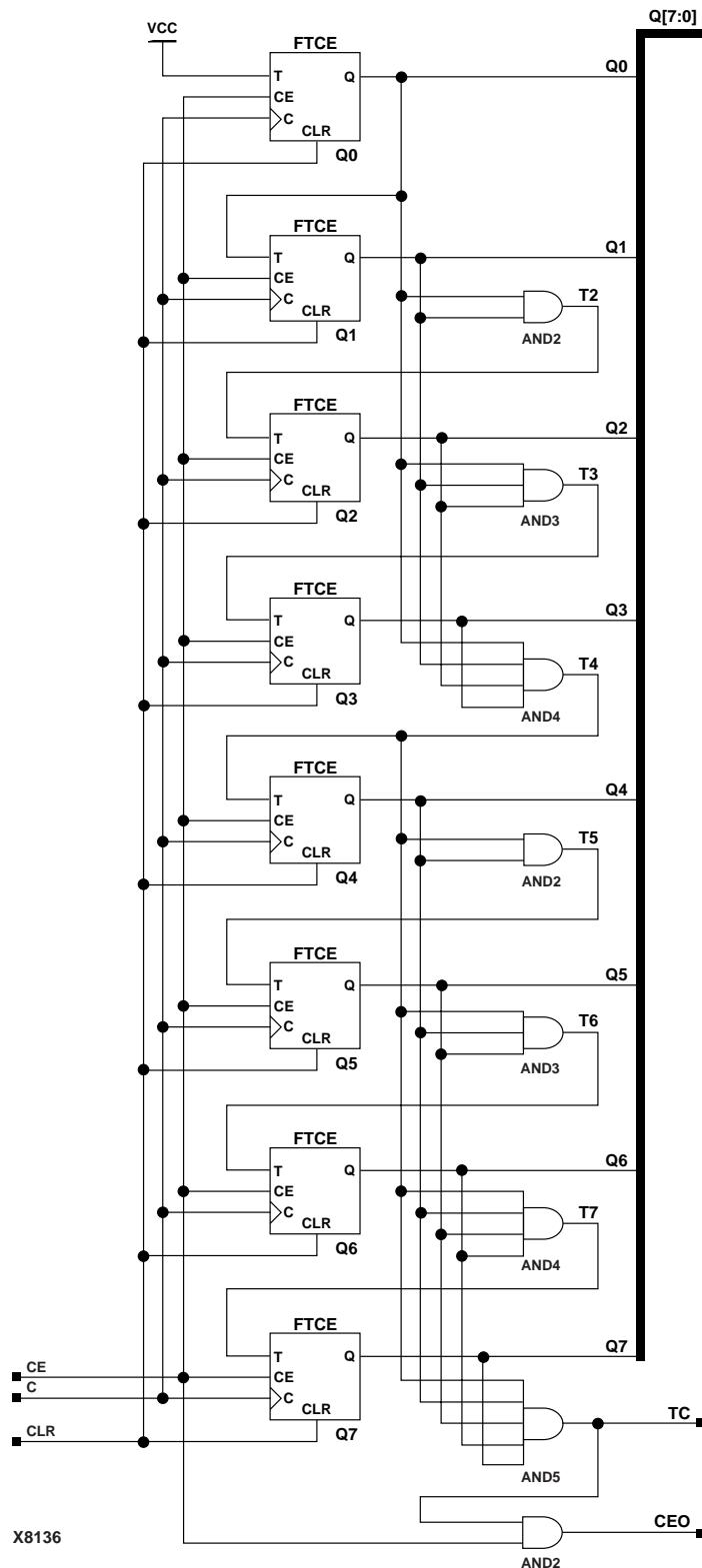
Inputs			Outputs		
CLR	CE	C	Qz-Q0	TC	CEO
1	X	X	0	0	0
0	0	X	No Chg	No Chg	0

Inputs			Outputs		
CLR	CE	C	Qz-Q0	TC	CEO
0	1	↑	Inc	TC	CEO

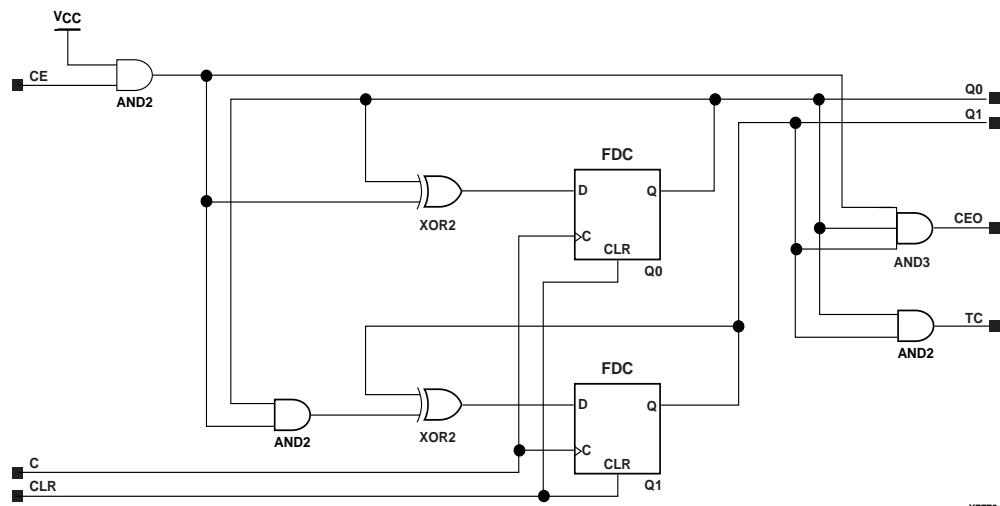
z= 1 for CB2CE; z = 3 for CB4CE; z = 7 for CB8CE; z = 15 for CB16CE

TC = Qz•Q(z-1)•Q(z-2)•...•Q0

CEO = TC•CE

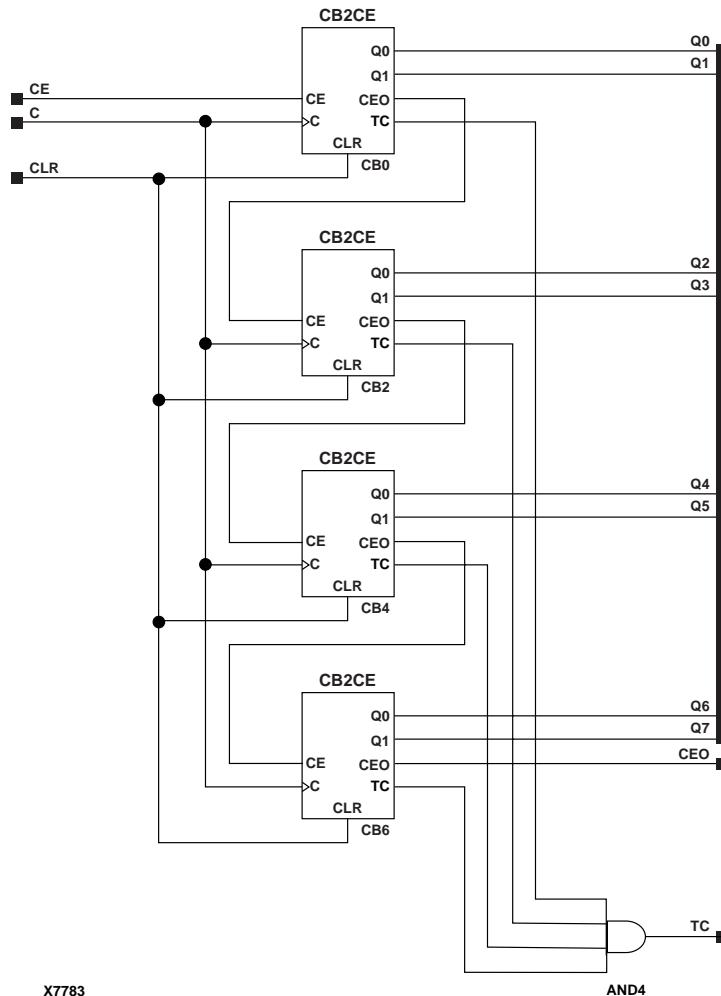


**CB8CE Implementation Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E,**

**Virtex-II, Virtex-II Pro, Virtex-II Pro X**

X7779

**CB2CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II**



### CB8CE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

## Usage

For HDL, these design elements are inferred rather than instantiated.

## VHDL Inference Code

```

architecture Behavioral of cb2ce is

constant TERMINAL_COUNT : std_logic_vector(WIDTH-1 downto 0) :=
(others => '1');

begin

process(C, CLR)
begin
  if (CLR='1') then
    Q <= (others => '0');
  elsif (C'event and C='1') then

```

```
    if (CE='1') then
        Q <= Q+1;
    end if;
end if;
end process;

process (Q)
begin
    if (Q = TERMINAL_COUNT) then
        TC <='1';
    else
        TC <='0';
    end if;
end process;

CEO<=TC and CE;

end Behavioral;
```

### Verilog Inference Code

```
always @ (posedge C or posedge CLR)
begin
    if (CLR)
        Q <= 0;
    else if (CE)
        Q <= Q + 1;
end

always @ (Q)
begin
    if (Q == TERMINAL_COUNT)
        TC <= 1;
    else
        TC <= 0;
end

always @ (CE or TC)
begin
    CEO <= TC && CE;
end
```