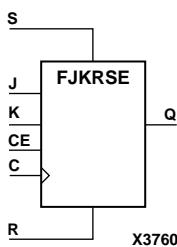


FJKRSE

J-K Flip-Flop with Clock Enable and Synchronous Reset and Set

Architectures Supported

FJKRSE	
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



FJKRSE is a single J-K-type flip-flop with J, K, synchronous reset (R), synchronous set (S), and clock enable (CE) inputs and data output (Q). When synchronous reset (R) is High, all other inputs are ignored and output Q is reset Low. (Reset has precedence over Set.) When synchronous set (S) is High and R is Low, output Q is set High. When R and S are Low and CE is High, output Q responds to the state of the J and K inputs, according to the following truth table, during the Low-to-High clock (C) transition. When CE is Low, clock transitions are ignored.

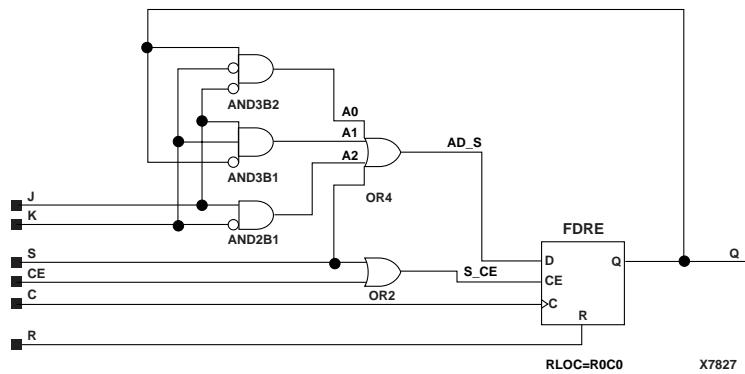
The flip-flop is asynchronously cleared, output Low, when power is applied.

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.

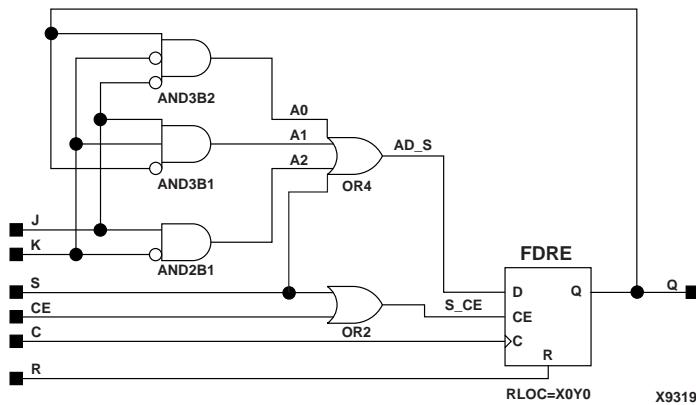
Spartan-II, Spartan-IIIE, Spartan-3, Virtex, Virtex-E, Virtex-II, Virtex-II Pro, and Virtex-II Pro X simulate power-on when global set/reset (GSR) is active.

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.

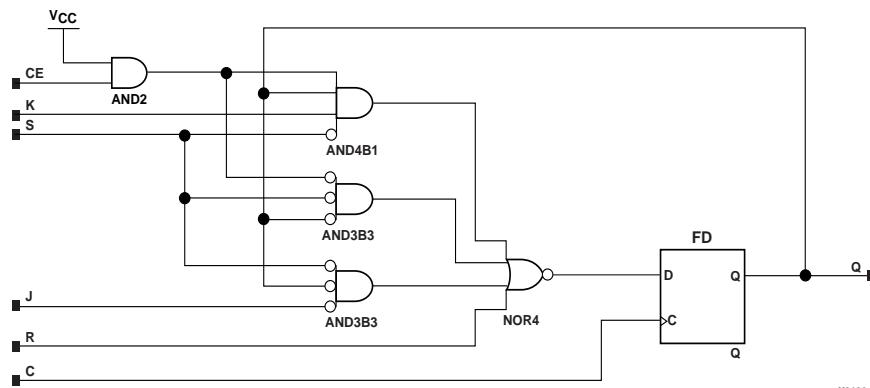
Inputs						Outputs
R	S	CE	J	K	C	Q
1	X	X	X	X	↑	0
0	1	X	X	X	↑	1
0	0	0	X	X	X	No Chg
0	0	1	0	0	X	No Chg
0	0	1	0	1	↑	0
0	0	1	1	1	↑	Toggle
0	0	1	1	0	↑	1



FJKRSE Implementation Spartan-II, Spartan-IIIE, Virtex, Virtex-E



FJKRSE Implementation Spartan-3, Virtex-II, Virtex-II Pro



FJKRSE Implementation XC9500/XV/XL, CoolRunner XPLA3, CoolRunner-II

Usage

For HDL, this design element is inferred rather than instantiated.

VHDL Inference Code

```

architecture Behavioral of fjkrsa is

begin
process (C)
begin
    if (C'event and C='1') then
        if (R='1') then
            Q <= '0';
        elsif (S='1') then
            Q <= '1';
        elsif (CE='1') then
            if (J='0') then
                if (K='1') then
                    Q <= '0';
                end if;
            else
                if (K='0') then
                    Q <= '1';
                else
                    Q <= not Q;
                end if;
            end if;
        end if;
    end if;
end process;

end Behavioral;

```

Verilog Inference Code

```

always @(posedge C)
begin
    if (R)
        Q <= 0;
    else if (S)
        Q <= 1;
    else if (CE)
        begin
            if (!J)
                begin
                    if (K)
                        Q <= 0;
                end
            else
                begin
                    if (!K)
                        Q <= 1;
                    else
                        Q <= !Q;
                end
        end
    end
end

```

