**FIFO code**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity fifo is

Port (

clk : in STD\_LOGIC;

reset : in STD\_LOGIC;

WriteEn : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR(1 downto 0);

ReadEn : in STD\_LOGIC;

DataOut : out STD\_LOGIC\_VECTOR(1 downto 0);

Empty : out STD\_LOGIC;

Full : out STD\_LOGIC

);

end fifo;

architecture Behavioral of fifo is

constant FIFO\_DEPTH : natural := 4; -- example depth

type FIFO\_Memory is array (0 to FIFO\_DEPTH-1) of STD\_LOGIC\_VECTOR(1 downto 0);

signal Memory : FIFO\_Memory := (others => (others => '0'));

signal Head : natural range 0 to FIFO\_DEPTH - 1 := 0;

signal Tail : natural range 0 to FIFO\_DEPTH - 1 := 0;

signal Looped : boolean := false;

signal dataOutReg : STD\_LOGIC\_VECTOR(1 downto 0) := (others => '0');

begin

DataOut <= dataOutReg;

fifo\_proc: process(clk, reset)

begin

if reset = '1' then

Head <= 0;

Tail <= 0;

Looped <= false;

dataOutReg <= (others => '0');

Full <= '0';

Empty <= '1';

elsif rising\_edge(clk) then

-- Read operation

if ReadEn = '1' then

if (Looped = true) or (Head /= Tail) then

dataOutReg <= Memory(Tail);

if Tail = FIFO\_DEPTH - 1 then

Tail <= 0;

Looped <= false;

else

Tail <= Tail + 1;

end if;

end if;

end if;

-- Write operation

if WriteEn = '1' then

if (not Looped) or (Head /= Tail) then

Memory(Head) <= DataIn;

if Head = FIFO\_DEPTH - 1 then

Head <= 0;

Looped <= true;

else

Head <= Head + 1;

end if;

end if;

end if;

-- Update flags

if Head = Tail then

if Looped then

Full <= '1';

Empty <= '0';

else

Empty <= '1';

Full <= '0';

end if;

else

Empty <= '0';

Full <= '0';

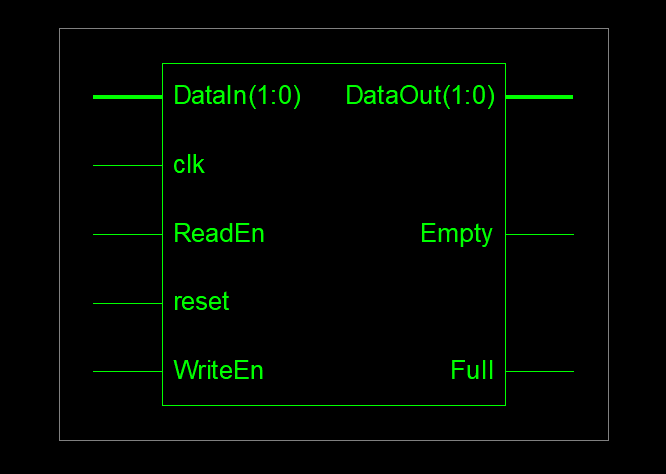
end if;

end if;

end process;

end Behavioral;

**RTL Design:**

****

Total memory usage is 208024 kilobytes

Number of errors : 0 ( 0 filtered)

Number of warnings : 0 ( 0 filtered)

Number of infos : 1 ( 0 filtered)