

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';
        end if;
    end process;
```

```

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;

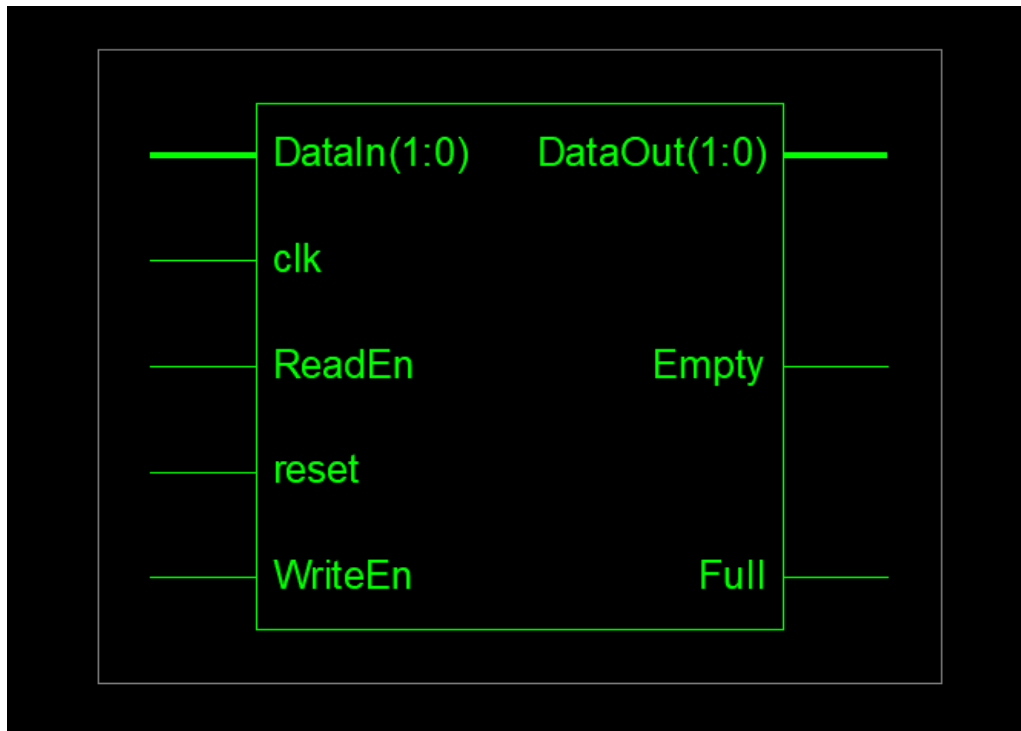
    -- Update flags
    if Head = Tail then
        if Looped then
            Full <= '1';
            Empty <= '0';
        else
            Empty <= '1';
            Full <= '0';
        end if;
    else
        Empty <= '0';
        Full <= '0';
    end if;

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)