ENTITY ASMMManualAssign IS
  PORT ( Clock, Resetn, InBit, BufFull : IN  STD_LOGIC ;
         state : BUFFER STD_LOGIC_Vector (1 DOWNTO 0);
         CountEN, RegLD, OutFlag : OUT  STD_LOGIC ) ;
END ASMMManualAssign ;

ARCHITECTURE ASMArch OF ASMMManualAssign IS
  SIGNAL StateA, StateB, StateC, StateD : STD_LOGIC ;
BEGIN
  PROCESS ( Resetn, Clock ) -- state transitions
  BEGIN
    IF Resetn = '0' THEN
      state <= "10" ;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      CASE state IS
        WHEN "10" =>
          IF InBit = '0' THEN state <= "11" ;
          ELSE state <= "00" ;
          END IF ;
          WHEN "11" =>
            state <= "10" ;
            WHEN "11" =>
              state <= "10" ;
  END PROCESS ;
WHEN "00" =>
    IF BufFull = '0' THEN state <= "00";
    ELSE state <= "01";
    END IF ;
WHEN "01" =>
    state <= "10";
WHEN OTHERS =>
    state <= "10";
END CASE ;
END IF ;
END PROCESS ;

-- Conditional and unconditional outputs
StateA <= '1' WHEN state = "10" ELSE '0';
StateB <= '1' WHEN state = "11" ELSE '0';
StateC <= '1' WHEN state = "00" ELSE '0';
StateD <= '1' WHEN state = "01" ELSE '0';

CountEN <= StateA OR (StateC AND BufFull);
RegLD <= StateA OR StateC;
OutFlag <= StateD;
END ASMArch ;
ENTITY ASMManualAssignGood IS
  PORT (  Clock, Resetn, InBit, BufFull : IN  STD_LOGIC ;
          state      :  BUFFER STD_LOGIC_Vector (1 DOWNTO 0);
          CountEN, RegLD, OutFlag  : OUT  STD_LOGIC ) ;
END ASMManualAssignGood ;

ARCHITECTURE ASMArch OF ASMManualAssignGood IS
  CONSTANT A : STD_LOGIC_Vector (1 DOWNTO 0):= "10";
  CONSTANT B : STD_LOGIC_Vector (1 DOWNTO 0):= "11";
  CONSTANT C : STD_LOGIC_Vector (1 DOWNTO 0):= "00";
  CONSTANT D : STD_LOGIC_Vector (1 DOWNTO 0):= "01";
BEGIN
  PROCESS ( Resetn, Clock ) -- State transitions
  BEGIN
    IF Resetn = '0' THEN
      state <= A ;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      CASE state IS
        WHEN A =>
          IF InBit = '0' THEN state <= B ;
          ELSE state <= C ;
          END IF ;
        WHEN B =>
          state <= A ;
        WHEN C =>
          IF BufFull = '0' THEN state <= C ;
          ELSE state <= D ;
      END CASE ;
    END IF ;
  END PROCESS ;
END ASMArch ;
END IF;
WHEN D =>
    state <= A;
WHEN OTHERS =>
    state <= A;
END CASE;
END IF;
END PROCESS;

PROCESS (state, BufFull) -- conditional and uncond. outputs
BEGIN    -- All outputs are default to '0'
    CountEN <= '0';
    RegLD <= '0';
    OutFlag <= '0';

    CASE state IS
        WHEN A =>
            CountEN <= '1';
            RegLD <= '1';
        WHEN C =>
            RegLD <= '1';
            IF BufFull = '1' THEN CountEN <= '1';
        END IF;
        WHEN D =>
            OutFlag <= '1';
        WHEN OTHERS =>
    END CASE;
END PROCESS;

END ASMArch;
ASMManualAssignBetter.vhd
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ASMManualAssignBetter IS
  PORT ( Clock, Resetn, InBit, BufFull        : IN  STD_LOGIC;
         state      : BUFFER STD_LOGIC_VECTOR (1 DOWNTO 0);
         CountEN, RegLD, OutFlag  : OUT  STD_LOGIC ) ;
END ASMManualAssignBetter ;

ARCHITECTURE ASMArch OF ASMManualAssignBetter IS
  CONSTANT A : STD_LOGIC_VECTOR (1 DOWNTO 0):= "10";
  CONSTANT B : STD_LOGIC_VECTOR (1 DOWNTO 0):= "11";
  CONSTANT C : STD_LOGIC_VECTOR (1 DOWNTO 0):= "00";
  CONSTANT D : STD_LOGIC_VECTOR (1 DOWNTO 0):= "01";

  SIGNAL nextState: STD_LOGIC_VECTOR (1 DOWNTO 0);

BEGIN
  PROCESS ( Resetn, Clock )  -- State transitions
  BEGIN
    IF Resetn = '0' THEN
      state <= A ;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      state <= nextState;
    END IF;
  END PROCESS ;
PROCESS (state, InBit, BufFull) -- conditional and uncond. Outputs, next state
BEGIN    -- All outputs are default to '0'
    CountEN <= '0';
    RegLD <= '0';
    OutFlag <= '0';

    CASE state IS
        WHEN A =>
            CountEN <= '1';
            RegLD <= '1';
            IF InBit = '0' THEN nextState <= B ;
            ELSE nextState <= C ;
            END IF ;
        WHEN B =>
            nextState <= A;
        WHEN C =>
            RegLD <= '1';
            IF BufFull = '1' THEN
                CountEN <= '1';
                nextState <= D;
            ELSE
                nextState <= C;
            END IF ;
        WHEN D =>
            OutFlag <= '1';
            nextState <= A;
        WHEN OTHERS =>
            nextState <= A;
    END CASE ;
END PROCESS ;
END ASMArch ;
ASMuserDefineType.vhd:

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ASMuserDefinedtype IS
  PORT (Clock, Resetn, InBit, BufFull : IN STD_LOGIC;
        CountEN, RegLD, OutFlag : OUT STD_LOGIC);
END ASMuserDefinedtype;

ARCHITECTURE ASMArch OF ASMuserDefinedtype IS
  TYPE ASMstateType IS (A, B, C, D); -- User defined signal type
  SIGNAL state : ASMstateType; -- the signal "state" is of the type "ASMstateType"

BEGIN
  PROCESS (Resetn, Clock) -- State transitions
  BEGIN
    IF Resetn = '0' THEN
      state <= A; -- Note that “A” is the actual value.
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      state <= nextState;
    END IF;
  END PROCESS;
END ASMArch;
PROCESS (state, InBit, BufFull) -- conditional and uncond. Outputs, next state
BEGIN -- All outputs are default to '0'
    CountEN <= '0';
    RegLD <= '0';
    OutFlag <= '0';

    CASE state IS
        WHEN A =>
            CountEN <= '1';
            RegLD <= '1';
            IF InBit = '0' THEN nextState <= B ;
            ELSE nextState <= C ;
            END IF ;
        WHEN B =>
            nextState <= A;
        WHEN C =>
            RegLD <= '1';
            IF BufFull = '1' THEN
                CountEN <= '1';
                nextState <= D;
            ELSE
                nextState <= C;
            END IF ;
        WHEN D =>
            OutFlag <= '1';
            nextState <= A;
        WHEN OTHERS =>
            nextState <= A;
    END CASE ;
END PROCESS ;
END ASMArch ;