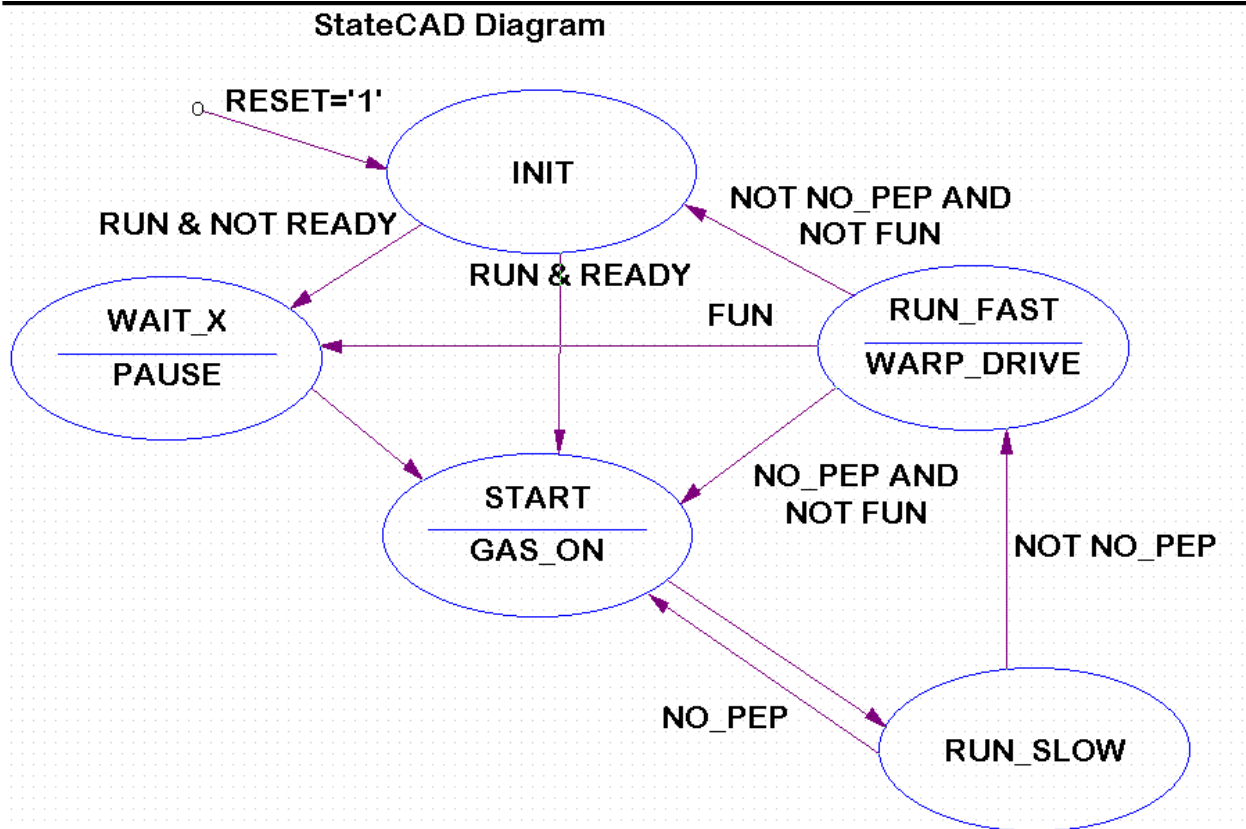


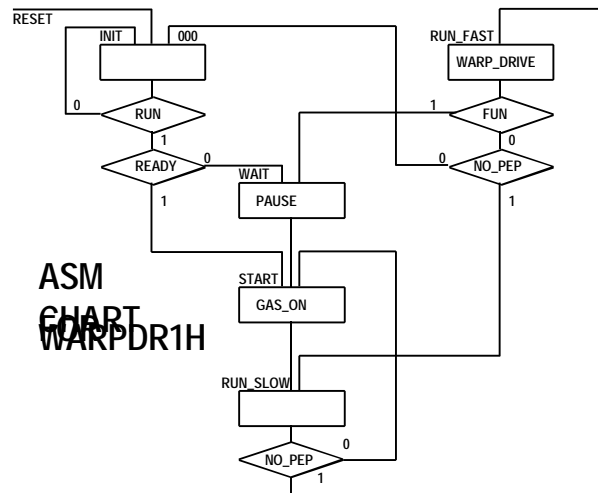
LIFE WITHOUT STATECAD  
OR  
YES, IT CAN BE DONE!

If you do not use StateCAD in the design of a FSM, there are other ways of doing the design. It is just a bit more work. The following is an example and a “template” which you may use.

Consider the FSM described the state diagram below.



The first step is optional. This is to convert this diagram to an ASM chart. The ASM chart contains the same information - it is just drawn differently. The chart for this machine is shown to the right. Circles are replaced with rectangles. State names are placed outside the state (upper left-hand corner). Decision boxes are used to indicate the next states. It is the decision boxes that make writing HDL code a bit easier. It can be written directly from the state diagram. (Or, you can skip the state diagram and draw the ASM chart directly).



You must decide whether to use **binary encoded** or **one-hot** state assignment. The you can convert the ASM chart (or state diagram) to VHDL by using the form below. The *italicized* text is what you have to add from the ASM chart or state diagram (this is binary encoded).

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY WARPDRVE IS
    PORT (CLK, RESET, other inputs      : IN std_logic;
          outputs                       : OUT std_logic);
END;
ARCHITECTURE BEHAVIOR OF WARPDRVE IS
    SIGNAL sreg      : std_logic_vector (N-1 DOWNTO 0);1
    SIGNAL next_sreg : std_logic_vector (N-1 DOWNTO 0);
    CONSTANT FIRST_STATE_NAME : std_logic_vector (N-1 DOWNTO 0) := "bbb";2
    . . .
    CONSTANT LAST_STATE_NAME : std_logic_vector (N-1 DOWNTO 0) := "bbb";
BEGIN
    PROCESS (CLK, RESET, next_sreg)
    BEGIN
        IF ( RESET='1' ) THEN sreg <= INIT;
        ELSIF CLK='1' AND CLK'event THEN sreg <= next_sreg;
        END IF;
    END PROCESS;

    PROCESS (sreg, inputs other than clock and reset)
    BEGIN
        FIRST_STATE_NAME <= '0'; . . .
        LAST_STATE_NAME  <= '0';

        next_sreg<= INITIAL STATE;

        CASE sreg IS
            WHEN STATE_NAME =>
                FIRST_OUTPUT <= '0';
                . . .
                LAST_OUTPUT <= '0';
                IF ( FIRST_CONDITION='0' ) THEN
                    next_sreg<=THAT_STATE;
                END IF;
                IF ( FIRST_CONDITION='1' AND ANOTHER_CONDITION='0' )
                THEN next_sreg<=ANOTHER_STATE;
                END IF;
                IF ETC. . . . .
                END IF;
            WHEN ANOTHER_STATE =>
                ETC.
            WHEN OTHERS =>
                . . .
        END CASE;
    END PROCESS;
END BEHAVIOR;

```

If you choose to use one-hot encoding, the form is as below.

---

<sup>1</sup>N is the number of state variables needed

<sup>2</sup>bbb is the state assignment for the names state

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY WARPDR1H IS
    PORT (CLK, FUN, OTHER INPUTS: IN std_logic;
          OUTPUTS: OUT std_logic;
          STATE_NAMES: BUFFER std_logic);
END;
ARCHITECTURE BEHAVIOR OF WARPDR1H IS
-- State variables for machine sreg
    SIGNAL next_FIRST_STATE_NAME . . . , next_LAST_STATE_NAME :
        std_logic;
BEGIN
    PROCESS (CLK, RESET, next_FIRST_STATE_NAME . . . ,
            next_LAST_STATE_NAME)
    BEGIN
        IF ( RESET='1' ) THEN
            FIRST_STATE_NAME <= '0';
            . . . ;
            LAST_STATE_NAME <= '0';
            ELSIF CLK='1' AND CLK'event THEN
                FIRST_STATE_NAME <= next_FIRST_STATE_NAME;
                . . .
                LAST_STATE_NAME <= next_LAST_STATE_NAME;
            END IF;
        END PROCESS;

        PROCESS (INPUTS, STATE_NAMES)
        BEGIN

            IF ( SOME_CONDITIONS LEADING TO THIS_STATE )
            THEN THIS_STATE<='1';
            ELSE THIS_STATE<='0';
            END IF;

            IF ( DO THE SAME FOR THE OTHER STATES . . .

            IF (( (SOME_STATE_NAME='1'))
            THEN SOME_OUTPUT<='1';
            ELSE SOME_OUTPUT<='0';
            END IF;

            IF ( REPEAT FOR OTHER STATES AND OUTPUTS . . .

        END PROCESS;
    END BEHAVIOR;

```

These templates should be of some help in doing VHDL by hand. You will still have to give some thought to what is to be inserted in the *italicized* sections.

The actual VHDL code (generated by StateCAD for this machine is shown below (both forms).

```

-- This code is to be used for evaluation purposes only
-- F: \PROJS319\MANUAL\WARPDRVE.vhd

```

```
-- VHDL code created by Visual Software Solution's StateCAD Version 4.0
-- Tue Apr 14 10:17:57 1998

-- This VHDL code (for use with IEEE compliant tools) was generated using:
-- binary encoded state assignment with structured code format.
-- Minimization is enabled, implied else is enabled,
-- and outputs are area optimized.
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
ENTITY WARPDRVE IS
    PORT (CLK, FUN, NO_PEP, READY, RESET, RUN: IN std_logic;
          GAS_ON, PAUSE, WARP_DRIVE : OUT std_logic);
END;
```

```
ARCHITECTURE BEHAVIOR OF WARPDRVE IS
    SIGNAL sreg : std_logic_vector (2 DOWNTO 0);
    SIGNAL next_sreg : std_logic_vector (2 DOWNTO 0);
    CONSTANT INIT : std_logic_vector (2 DOWNTO 0) := "000";
    CONSTANT RUN_FAST : std_logic_vector (2 DOWNTO 0) := "001";
    CONSTANT RUN_SLOW : std_logic_vector (2 DOWNTO 0) := "010";
    CONSTANT START : std_logic_vector (2 DOWNTO 0) := "011";
    CONSTANT WAIT_X : std_logic_vector (2 DOWNTO 0) := "100";
```

```
BEGIN
    PROCESS (CLK, RESET, next_sreg)
    BEGIN
        IF ( RESET='1' ) THEN
            sreg <= INIT;
        ELSIF CLK='1' AND CLK'event THEN
            sreg <= next_sreg;
        END IF;
    END PROCESS;

    PROCESS (sreg, FUN, NO_PEP, READY, RUN)
    BEGIN
        GAS_ON <= '0'; PAUSE <= '0'; WARP_DRIVE <= '0';

        next_sreg<=INIT;

        CASE sreg IS
            WHEN INIT =>
                WARP_DRIVE<='0';
                PAUSE<='0';
                GAS_ON<='0';
                IF ( RUN='0' ) THEN
                    next_sreg<=INIT;
                END IF;
                IF ( RUN='1' AND READY='0' ) THEN
                    next_sreg<=WAIT_X;
                END IF;
                IF ( RUN='1' AND READY='1' ) THEN
                    next_sreg<=START;
                END IF;
            WHEN RUN_FAST =>
                PAUSE<='0';
                GAS_ON<='0';
                WARP_DRIVE<='1';
                IF ( FUN='1' ) THEN
                    next_sreg<=WAIT_X;
                END IF;
                IF ( NO_PEP='0' AND FUN='0' ) THEN
                    next_sreg<=INIT;
                END IF;
        END CASE;
    END PROCESS;
END;
```

```

        END IF;
        IF ( NO_PEP=' 1' AND FUN=' 0' ) THEN
            next_sreg<=START;
        END IF;
    WHEN RUN_SLOW =>
        WARP_DRIVE<=' 0' ;
        PAUSE<=' 0' ;
        GAS_ON<=' 0' ;
        IF ( NO_PEP=' 0' ) THEN
            next_sreg<=RUN_FAST;
        END IF;
        IF ( NO_PEP=' 1' ) THEN
            next_sreg<=START;
        END IF;
    WHEN START =>
        WARP_DRIVE<=' 0' ;
        PAUSE<=' 0' ;
        GAS_ON<=' 1' ;
        next_sreg<=RUN_SLOW;
    WHEN WAIT_X =>
        WARP_DRIVE<=' 0' ;
        GAS_ON<=' 0' ;
        PAUSE<=' 1' ;
        next_sreg<=START;
    WHEN OTHERS =>
        END CASE;
    END PROCESS;
END BEHAVIOR;

```

---

```

-- This code is to be used for evaluation purposes only
-- F: \PROJS319\MANUAL\WARPD1H.vhd
-- VHDL code created by Visual Software Solution's StateCAD Version 4.0
-- Tue Apr 14 10:16:41 1998

```

```

-- This VHDL code (for use with IEEE compliant tools) was generated using:
-- one-hot state assignment with boolean code format.
-- Minimization is enabled, implied else is enabled,
-- and outputs are area optimized.

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

```

```

ENTITY WARPD1H IS
    PORT (CLK, FUN, NO_PEP, READY, RESET, RUN: IN std_logic;
          GAS_ON, PAUSE, WARP_DRIVE : OUT std_logic;
          INIT, RUN_FAST, RUN_SLOW, START, WAIT_X : BUFFER std_logic);
END;

```

```

ARCHITECTURE BEHAVIOR OF WARPD1H IS

```

```

-- State variables for machine sreg
    SIGNAL next_INIT, next_RUN_FAST, next_RUN_SLOW, next_START,
           next_WAIT_X : std_logic;
BEGIN
    PROCESS (CLK, RESET, next_INIT, next_RUN_FAST, next_RUN_SLOW,
            next_START, next_WAIT_X)
    BEGIN
        IF ( RESET=' 1' ) THEN
            INIT <= ' 1' ;
            RUN_FAST <= ' 0' ;
            RUN_SLOW <= ' 0' ;
            START <= ' 0' ;
            WAIT_X <= ' 0' ;

```

```

    ELSIF CLK='1' AND CLK'event THEN
        INIT <= next_INIT;
        RUN_FAST <= next_RUN_FAST;
        RUN_SLOW <= next_RUN_SLOW;
        START <= next_START;
        WAIT_X <= next_WAIT_X;
    END IF;
END PROCESS;

PROCESS (FUN, INIT, NO_PEP, READY, RUN, RUN_FAST, RUN_SLOW, START, WAIT_X)
BEGIN

    IF (( RUN='0' AND (INIT='1')) OR ( NO_PEP='0' AND FUN='0'
    AND (RUN_FAST='1'))))
    THEN next_INIT<='1';
    ELSE next_INIT<='0';
    END IF;

    IF (( NO_PEP='0' AND (RUN_SLOW='1'))))
    THEN next_RUN_FAST<='1';
    ELSE next_RUN_FAST<='0';
    END IF;

    IF (( (START='1'))))
    THEN next_RUN_SLOW<='1';
    ELSE next_RUN_SLOW<='0';
    END IF;

    IF (( RUN='1' AND READY='1' AND (INIT='1'))
    OR ( NO_PEP='1' AND FUN='0' AND (RUN_FAST='1'))
    OR ( NO_PEP='1' AND (RUN_SLOW='1')) OR ( (WAIT_X='1'))))
    THEN next_START<='1';
    ELSE next_START<='0';
    END IF;

    IF (( RUN='1' AND READY='0' AND (INIT='1'))
    OR ( FUN='1' AND (RUN_FAST='1'))))
    THEN next_WAIT_X<='1';
    ELSE next_WAIT_X<='0';
    END IF;

    IF (( (START='1'))))
    THEN GAS_ON<='1';
    ELSE GAS_ON<='0';
    END IF;

    IF (( (WAIT_X='1'))))
    THEN PAUSE<='1';
    ELSE PAUSE<='0';
    END IF;

    IF (( (RUN_FAST='1'))))
    THEN WARP_DRIVE<='1';
    ELSE WARP_DRIVE<='0';
    END IF;
END PROCESS;
END BEHAVIOR;

```