

### § 3 Minimization Techniques and Digital System Design

#### § 3.1 Combinational Circuits Minimization Techniques

1. Much of the simplifying work was done to the datapath
  - a. Boolean Logic Theorems
  - b. K-Map (3-6 variables, more than that will be less useful)
  - c. Tabular Method (Based on Quine-McClusky method)
  - d. Use Software (Like Espresso)

2. Example:

$$F(A, B, C, D) = \Sigma (4, 5, 6, 8, 9, 10, 13)$$

$$d(A, B, C, D) = \Sigma (0, 7, 15)$$

- a. Arrange the minterms based on the hamming weight, that is, the number of 1's in the minterm

- b. Arrange in the table

- c. Find the essential groups, that is, group cover most variables

### § 3.2 Sequential Logic Circuits Minimization

Present State	Next State		Output ( $y_1 y_0$ )	
	Input $x = 0$	Input $x = 1$	Input $x = 0$	Input $x = 1$
<b>M</b>	<b>m</b>	<b>n</b>	<b>00</b>	<b>01</b>
<b>N</b>	<b>q</b>	<b>n</b>	<b>11</b>	<b>10</b>
<b>O</b>	<b>o</b>	<b>n</b>	<b>00</b>	<b>01</b>
<b>P</b>	<b>p</b>	<b>r</b>	<b>00</b>	<b>01</b>
<b>Q</b>	<b>s</b>	<b>r</b>	<b>11</b>	<b>00</b>
<b>R</b>	<b>q</b>	<b>r</b>	<b>11</b>	<b>10</b>
<b>S</b>	<b>o</b>	<b>r</b>	<b>00</b>	<b>01</b>

## § 3.2 Digital System Design Example: Serial to Parallel Converter (STOP)

### 1. Specification

- Synchronous reset (R)
- Input A is asserted for exactly one clock period prior to the arrival of serial data on input D
- For the next four clock cycles, data arrives serially on D
- Device collects four bits of serial data and output them in parallel at output Z
- When parallel data appears, signal DONE asserts
- Z and DONE must remain asserted for one full clock cycle
- During the clock cycle when the parallel data is at Z, the device may receive another pulse on A, indicating that new data will arrive
  - Yes, be prepared to receive
  - No, goes to reset state after sending out Z

### 2. Timing diagram

### 3. Step 1: Moore / Mealy Machine

- ➔ Functionally, it is always possible to build any specification as wither a Moore machines or a Mealy machine.
- ➔ Primary difference is in the timing of the outputs.

4. Three practical effects to consider:
- In Moore machine, outputs settle to their final values a few gate delays after the active clock edge. They are constant for the remainder of the clock period, even if inputs happen to change during the clock period.
    - It isolates outputs from inputs.
  - Mealy machine respond one clock period earlier than Moore machine to input changes, but also allows outputs to follow spurious input changes.
    - Noise on the input lines may be transferred to the outputs.
  - Moore machine may require more states than that from a Mealy machine.
- For the design example, output must be present during the clock cycle following the last input. Outputs required to be constant during the entire period, a Mealy machine can't be used.

5. Step 2: construct of state table: follow a structured methodology

- Two approaches:

(1) State diagrams

**Advantage:** simple, easy to start with

**Disadvantage:** limit to relatively small devices, but enough for STOP

→ Start with a state that is easily described in words. If there's a reset state, it is always good place to start with. Write a complete word description of each state as it's described.

→ **Reset State (S0)**

Entered at the end of any clock where  $R = 1$

Stays until  $A = 1$

DONE = 0, output Z is unspecified.

Then decide what to do in S0 for various conditions on inputs.

→ **State S1**

Entered from S0 when  $R = 0$  and  $A = 1$

Data on D must be saved at the clock edge for later output.

DONE = 0, output Z unspecified.

... ..

Repeat until the complete state diagram is achieved.

(2) Transition List Approach

**Advantage:** good for problems that are too complex for a state diagram to be constructed.

Present State	Condition	Next State	Data Transferred	Outputs
S0	$R + (A)$	S0	None	0-
S0	$(R).A$	S1	None	0-
S1	(R)	S2	Store Bit1	0-
S1	R	S0	None	0-
S2	(R)	S3	Store Bit2	0-
S2	R	S0	None	0-
S3	(R)	S4	Store Bit3	0-
S3	R	S0	None	0-
S4	(R)	S5	Store Bit4	0-
S4	R	S0	None	0-
S5	$(R).A$	S1	None	0-
S5	$R + (A)$	S0	None	0-

➔ Principle of mutual exclusion

➔ Used to aid the design process and check state diagrams for errors.

➔ Logic expressions on arcs leaving any node must be pair-wise mutually exclusive, that is, no two expressions on different arcs leaving the same node can be true simultaneously. E.g.,  $(R).A.[R + (A)] = 0$

➔ Diagram for circuit design

## 6. Step 3: VHDL coding

```
ENTITY stop IS
    PORT (R, A, D, CLK : IN BIT;
          Z : OUT BIT_VECTOR (3 DOWNTO 0);
          DONE : OUT BIT);
END stop;

ARCHITECTURE fsm_rtl OF stop IS
    TYPE state_type IS (S0, S1, S2, S3, S4, S5);
    SIGNAL state: state_type;
    SIGNAL shift_reg : BIT_VECTOR (3 DOWNTO 0);
BEGIN
    State_decoding: PROCESS (CLK)
        BEGIN
            IF (CLK = '0') THEN
                CASE state IS
                    WHEN S0 =>
                        IF (R='1' OR A='0') THEN
                            state <= S0;
                        ELSIF (R='0' OR A='1') THEN
                            state <= S1;
                        END IF;
                    WHEN S1 =>
                        shift_reg <= D & shift_reg (3 DOWNTO 1);
                        IF (R='0') THEN
                            state <= S2;
                        ELSIF (R='1') THEN
                            state <= S0;
                        END IF;
                    WHEN S2 =>
                        shift_reg <= D & shift_reg (3 DOWNTO 1);
                        IF (R='0') THEN
                            state <= S3;
                        ELSIF (R='1') THEN
                            state <= S0;
                        END IF;
                END CASE;
            END IF;
        END PROCESS;
END fsm_rtl;
```

```

        WHEN S3 =>
            shift_reg <= D & shift_reg (3 DOWNTO 1);
            IF (R='0') THEN
                state <= S4;
            ELSIF (R='1') THEN
                state <= S0;
            END IF;
        WHEN S4 =>
            shift_reg <= D & shift_reg (3 DOWNTO 1);
            IF (R='0') THEN
                state <= S5;
            ELSIF (R='1') THEN
                state <= S0;
            END IF;
        WHEN S5 =>
            IF (R='0' AND A='1') THEN
                state <= S1;
            ELSIF (R='1' OR A='0') THEN
                state <= S0;
            END IF;
    END CASE;
END IF;
END PROCESS state_decoding;

Output_proc: PROCESS (state)
BEGIN
    CASE state IS
        WHEN S0 TO S4 =>
            DONE <= '0';
        WHEN S5 =>
            DONE <= '1';
            Z <= shift_reg;
    END CASE;
END PROCESS output_proc;
END fsm_rtl;

```



### § 3.3 Implement a State Machine

Questions 1: Mealy machine or Moore machine?

Questions 2: Is this state diagram complete?

Questions 3: The timing diagram?

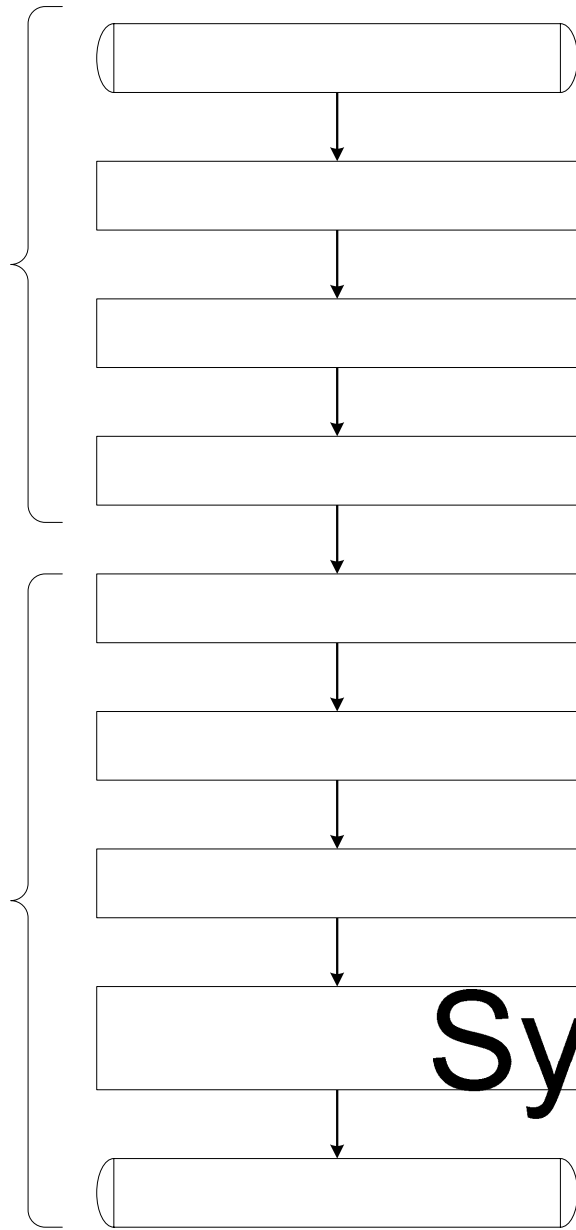
Questions 4: Realize with the fewest positive edge triggered T-FF?

Questions 5: What if D-FF will be used, or J-K FF? What if one hot design?

## § 3.4 Design Example (Microwave Oven Controller)

### § 3.5 ASIC Design Methodology

1. Generally speaking, Top-Down Design Process followed by Bottom-up Implementation
2. During the top-down design process, top level entity is recursively divided by using the divide-and-conquer strategy until all leaf components of the design tree become manageable.
3. Factors to be considered between a semi-custom ASIC device (FPGA, PLD) and a full-custom ASIC device (CMOS IC).
  - a. Speed
  - b. Number of input and output pins
  - c. Size of the circuit
    - i. Enough FFs / Latches / Gates
    - ii. Then actual mapping
  - d. If constraints from CLB and IOB and interconnections between blocks are not prohibitive  
→ ASIC might be good option if available
  - e. Number of gates < 1M
  - f. Number of pins < a few hundreds → available for FPGA
  - g. Volume of product
    - i. < hundreds of thousands → semi-custom
    - ii. > millions of devices → full-custom less expensive
    - iii. Prudent to first bring out an FPGA (less development time), in the mean time full-custom ASIC
  - h. If involves analog → full custom normally required
4. ASIC Design flow recommended by CMC



Synopsys