A SYSTEMATIC APPROACH TO THE DESIGN OF DIGITAL SYSTEMS

NOTES 2

A SYSTEMATIC APPROACH TO THE DESIGN OF DIGITAL SYSTEMS

In designing a digital system it is generally convenient to consider it as being composed of a controller and a processor as shown in Figure 1.

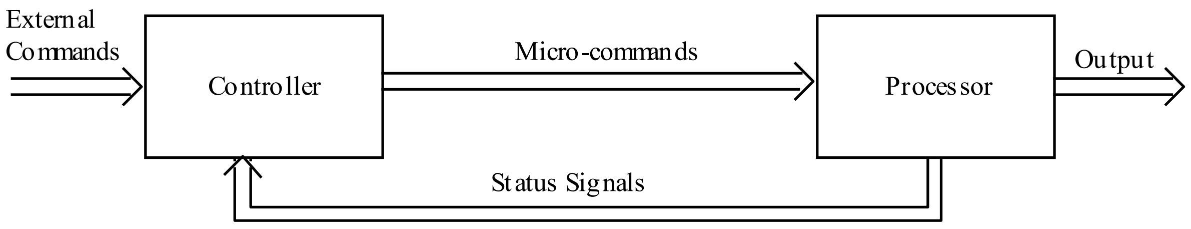

Figure 1. Control/Processor System Configuration

The controller and the processor can be designed relatively independently. The processor contains the registers, adders, counters, etc. which directly handle the data to be processed by the system or directly perform the desired system function. The controller is a sequential machine which receives external commands and status signals as its inputs and produces micro-commands which direct the operation of the processor.

A system of the form of Figure 1 is designed as follows:

(1) Decide on the functions to be performed in the processor and in the controller. Develop a block diagram similar to Figure 1 and indicate all external commands, micro-commands, and status signals to be used.

(2) Design the processor utilizing the signals as shown in (1).

(3) Design the controller to issue the required sequences of micro-commands.

It may not always be possible to obtain a "good" design by proceeding through the above three steps, without a certain amount of backtracking and modification.

The design procedure is best further explained by example.

EXAMPLE

Let us consider the design of a system which operates in the following manner. When an OP button is depressed, a 4-bit binary number is loaded from four switches. When the OP button is released, the "1" bits in the number are sequentially counted. If the count is odd, it is to be provided as the output. If the count is even, it is to be divided by 2 and the result is to be provided as the output. In addition, if the count was even an ALARM flag is to be set. The ALARM flag is to be cleared when the OP button is again depressed.

System Configuration

We will first propose a controller/processor split and associated set of signals for the system. The proposed configuration is shown in Figure 2.

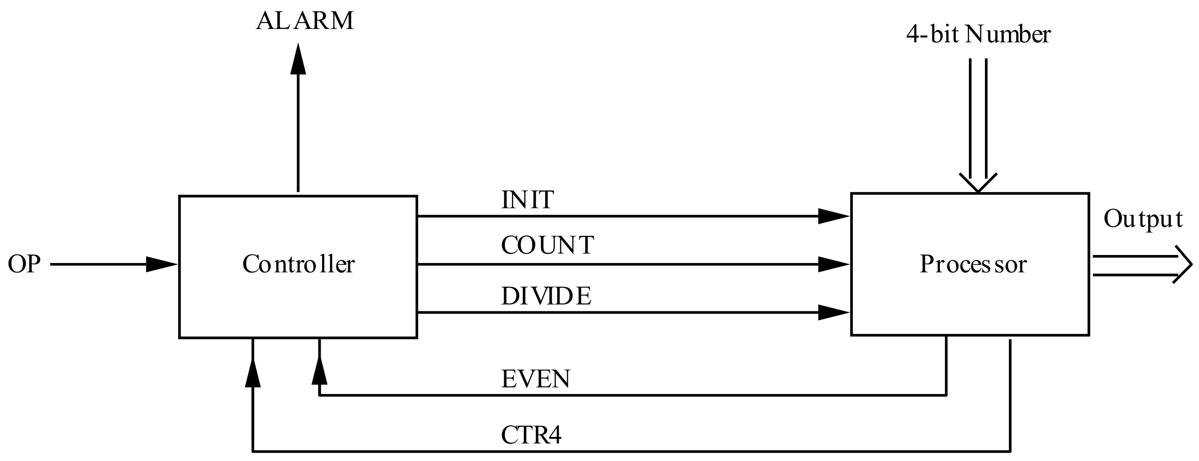

Figure 2. System Configuration

Signal Definitions:

INIT –	initializes the processor and loads and 4-bit number into the processor when high
COUNT –	the processor counts when high

DIVIDE – the count value is divided by two when high

EVEN – high when the count is even

CTR4 – high when all 4 bits of the number have been considered in the counting procedure and the counter value has been loaded into an "output register"

Processor Design

The major functions to be performed by the processor are:

(1) counting the number of 1's in the 4-bit number,

(2) dividing the result by 2.

(1) can be accomplished by incrementing the value of a counter (initialized as 0) each time the rightmost bit of the register containing the number is 1 while simultaneously shifting the number right in the register. (2) can be accomplished by shifting the count value one position to the right.

Note that an additional counter is necessary in the processor to count the number of bits of the 4-bit number which have been considered in the counting procedure so that the procedure can be terminated at the proper time. This counter must also be initialized as 0.

A processor design is given in Figure 3.

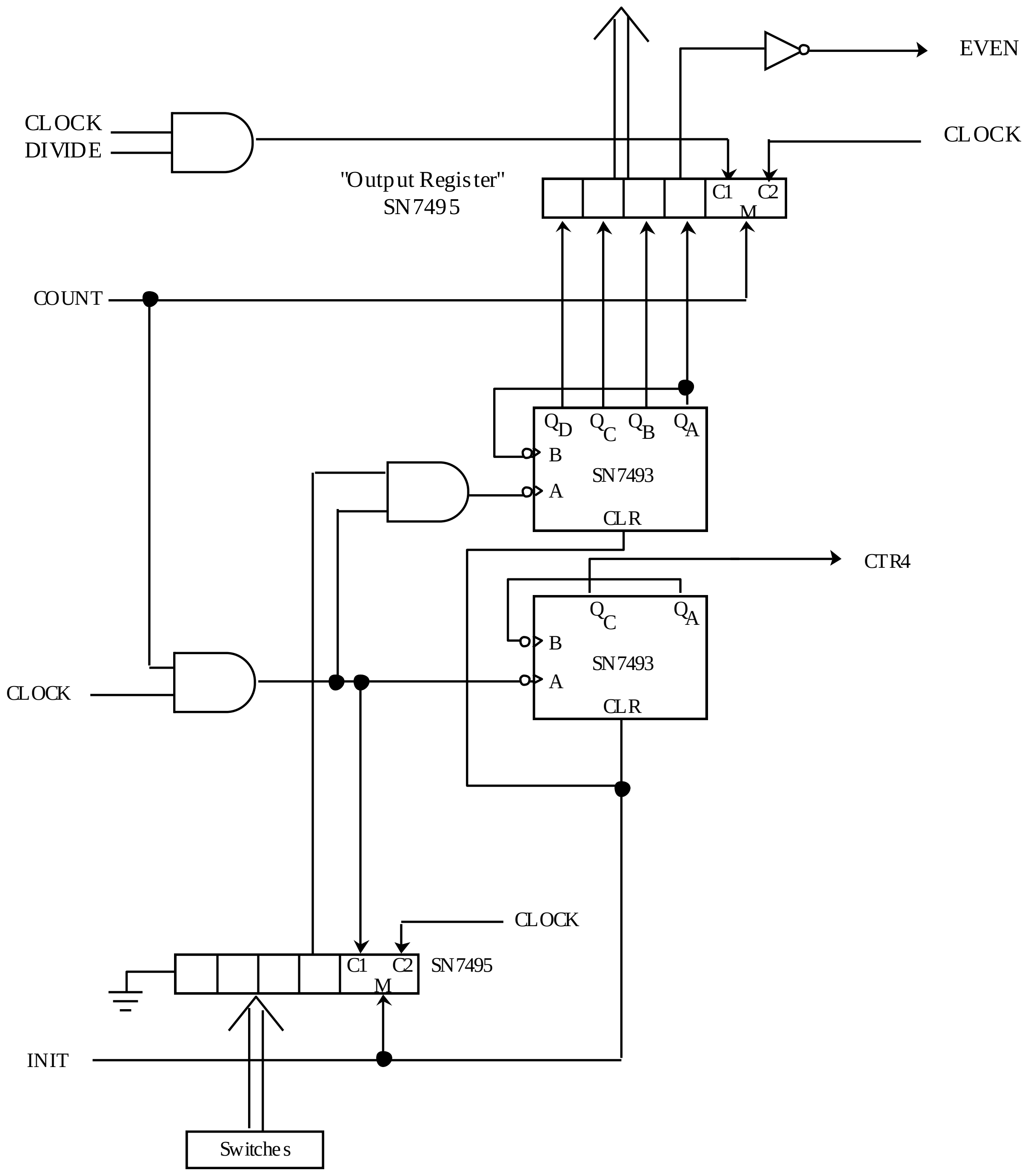

Figure 3. Processor Design

Controller Design

The controller is designed as a clocked sequential machine. We first develop a state-diagram describing the operation of the controller and then develop the flip-flop excitation functions and controller output functions from this diagram via Karnaugh maps.

Each state in our diagram is represented by a block as shown in Figure 4.

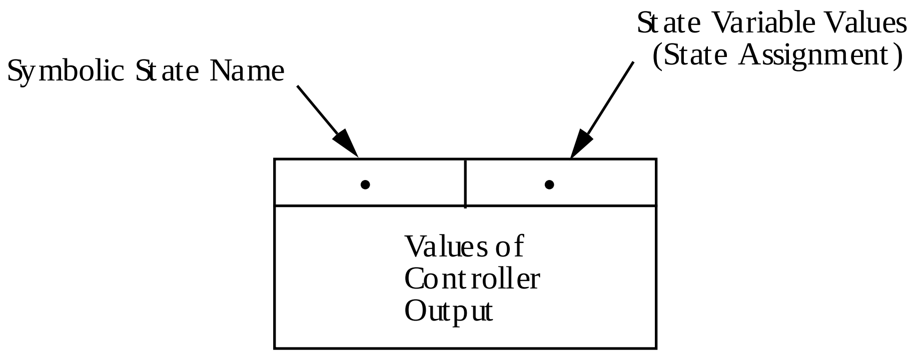

Figure 4. State Representation

Note that values of controller outputs can be expressed in the diagram as functions of external command signals and status signals as well as constants (0 or 1) or as "d" symbolizing "don't care." State transitions are represented by arrows. The Boolean functions which must be equal to 1 to initiate transitions are written beside the corresponding arrows. Self-loops (transitions from states back to themselves) are not explicitly indicated in the diagram.

A controller state-diagram for the example at hand is given in Figure 5.

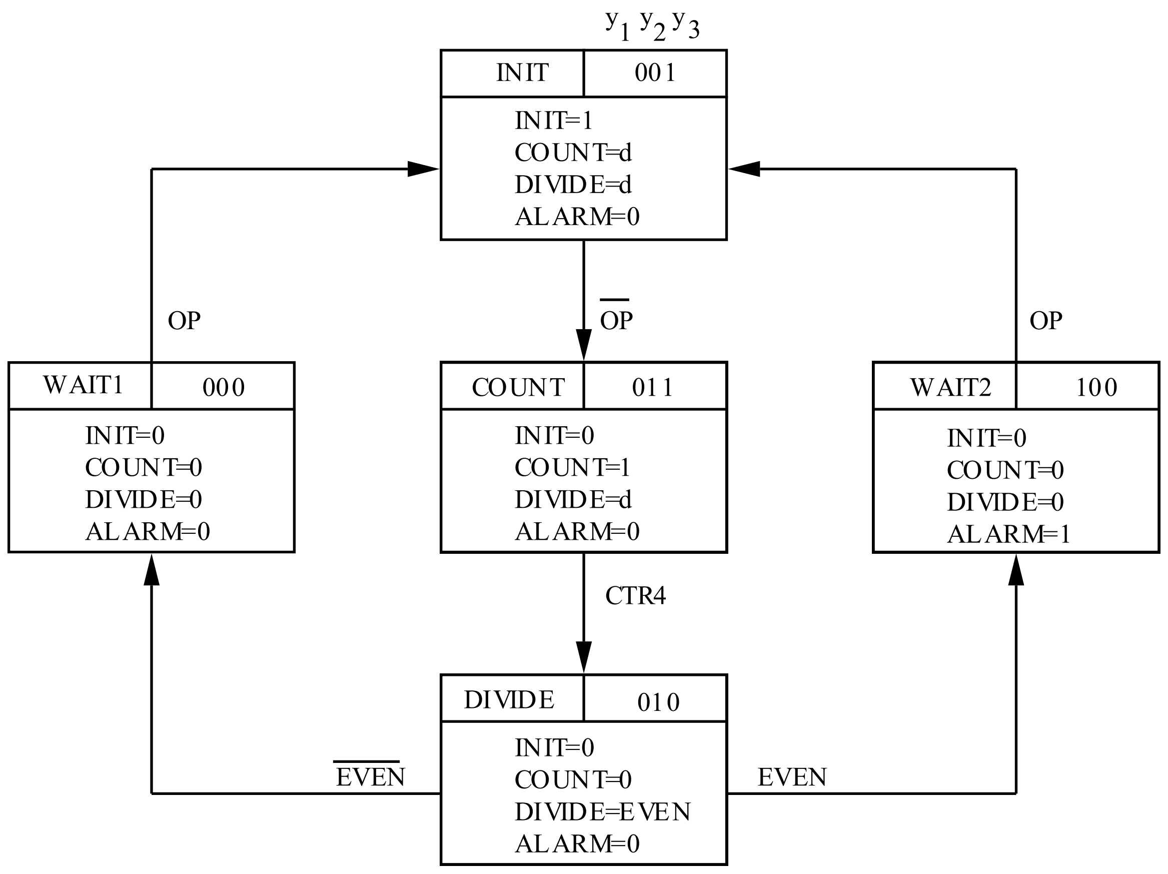

Figure 5. Controller State-diagram

Description

The system waits in either WAIT1 (ALARM off) or WAIT2 (ALARM on) until the OP button is depressed. In the INIT state, the number is loaded from the switches and the two counters and output register are cleared. After the OP button is released, the COUNT state is entered. In the COUNT state, the counting procedure is executed and a transition is made to DIVIDE after the result has been loaded into the output register. In the COUNT state, note that DIVIDE is a "don't care" since the C1 input to the output register (SN7495) is a "don't care" when the M (mode) input is high. Upon leaving the DIVIDE state, the result is shifted one position to the right if it was even and a transition is made to the appropriate WAIT state.

Karnaugh maps for the next-state functions and controller outputs are as given below.

$y_1 =$

$y_1 \backslash y_2y_3$	00	01	11	10
0	0	0	0	EVEN
1	OP	d	d	d

J_1

K_1

$y_2 =$

	0	$\overline{OP}$	1	0
y_1	0	d	d	d

J_2

K_2

$y_3 =$

	OP	1	$\overline{CTR4}$	0
y_1	OP	d	d	d

J_3

K_3

INIT=

	0	1	0	0
y_1	0	d	d	d

COUNT=

	0	d	1	0
y_1	0	d	d	d

DIVIDE=

	0	d	d	EVEN
y_1	0	d	d	d

ALARM=

	0	0	0	0
y_1	1	d	d	d

D.8

The excitation functions and input functions are read from the maps as follows:

$J_1 = y_2\overline{y}_3 \bullet \text{EVEN}$

$K_1 = \text{OP}$

$J_2 = y_3 \bullet \overline{\text{OP}}$

$K_2 = \overline{y}_3$

$J_3 = \overline{y}_2 \bullet \text{OP}$

$K_3 = y_2 \bullet \text{CTR 4}$

$\text{INIT} = \overline{y}_2 y_3$

$\text{COUNT} = y_3$

$\text{DIVIDE} = y_2\overline{y}_3 \bullet \text{EVEN}$

$\text{ALARM} = y_1$

The logic diagram for the controller is given in Figure 6.

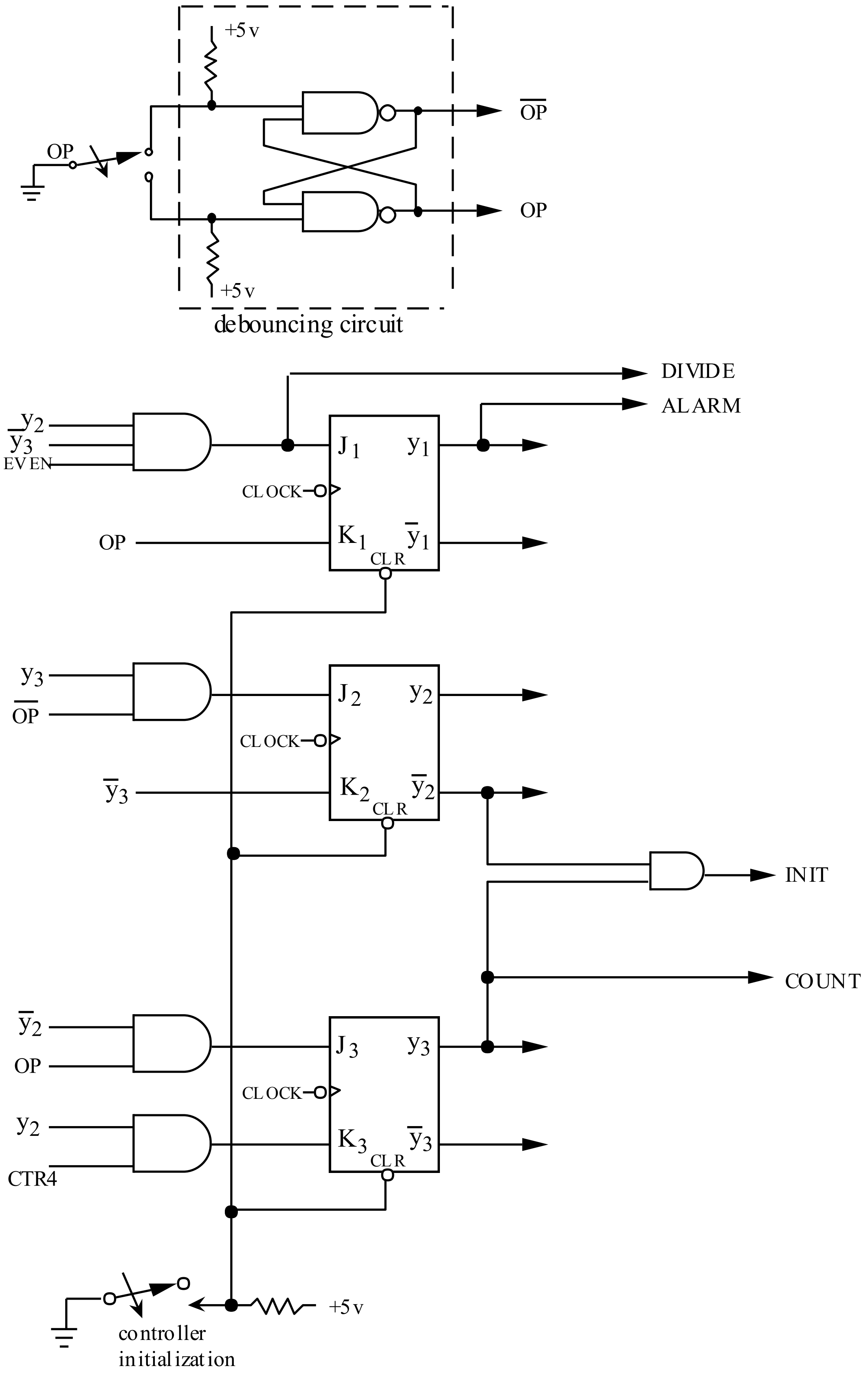

Figure 6. Controller Logic Diagram