Journal of Engineering and Applied Sciences 4 (5-6): 335-338, 2009
ISSN: 1816-949X
© Medwell Journals, 2009

Design of a Mealy State Digital Machine to Realised the Greatest Common Divisor (GCD) of Two Numbers

[1]D.A. Shalangwa and [2]D. Samaila
[1]Department of Physics, Adamawa State University,
[2]Department of Mathematics and Computer Science,
Adamawa State University, Mubi, Nigeria

Abstract: The desire in this research is to design a Mealy state digital machine using NAND gates only to compute the GCD of any two numbers. In the design, state selection, input variable, output variable definition and output assignment to ensure accurate result are considered. The method employed in this design is a classical method of digital design that utilized purely the State diagram, State Table, Excitation Table and Kanaugh Map (K-Map). The GCD of the numbers (40, 24) was obtained as 8. Although, the machine is capable of computing the GCD of any number provided the input is stated correctly.

Key words: GCD, HDL, K-Map, NAND gate, classical method

INTRODUCTION

A state digital machine transverses through a predetermined sequence of the state in an order fashion. A state is a set of values measured at different part of circuits. The digital machine designs are widely used for sequential control logic which forms the core of many digital systems (Papadinatou and Sarma, 1972), a state digital machine are required in a variety of applications covering a broad range of performance and complexity low level controls of microprocessor in VLSI peripheral interfaces, bus arbitration and timing generation in conventional microprocessor, custom bit slice microprocessor, data encryption and decryption and transmission protocol are but a few example (Hill and Peterson, 1997).

Typically, the details of control logic are the last to settle in the design circle, since they are continuously affected by changing system requirements and feature enhancement; programmable logic is a forgiving solution for control logic design because it allows easy modification to be made without disturbing PC board layout. Its flexibility provides an escape valve that permits design changes without impacting time market.

They are basically two methods of designing the state digital machine viz one hot method and classical method. In this research, classical method of design was chosen to design Mealy state machine to avoid erratic operation of circuitry driven by the state machine. The state machine architectures are shown in the Fig. 1.

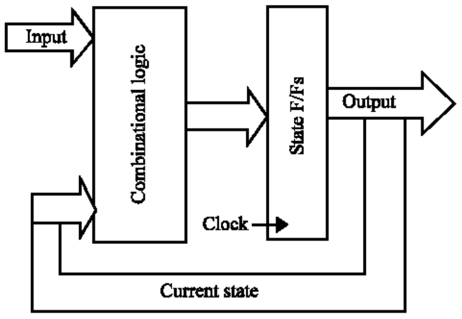


Fig. 1: Modified mealy digital machine

State diagram: State diagram allows the designer to describe the desired state machine operation graphically. This help to visualize the operation of the state machine prior to its implementation because it contains wealthy informations. The Fig. 1 shows state transition, the circles and the arrows describe, how the state machine moves from one state another. The circles represents a particular value of state variable, the arrows lines describe how the state machine transitions to the next state and arrow lines also contain the Boolean expression, which shows the criteria for a transition from one state to another. If the Boolean expression is true and the current state is the state at the source of the arrow line then the state machine will transit to the destination state on the clock (Fig. 2).

Input variable: All input variables were made to be synchronized to the state machine clock. If they are not, strange things will begin to happen in the actual operation

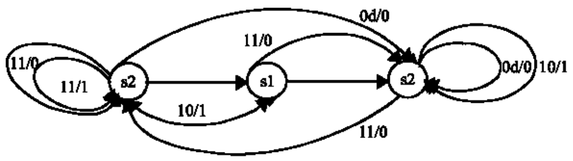


Fig. 2: State diagram of the Mealy digital machine (GCD) realize

(illegal state will be mysteriously entered), this could lead to total state machine lockup. Inevitably, the design will fail intermittently in the field (Hallbaueyer, 1998), this is because of the fact that a physical state machine implementation uses physical gates, which have non zero propagation delay. An input signal propagates through the gates to the D input of one state flip flop will slightly faster or slower than the same input signal travelling through a different set of gates to another flip flop's D input.

Output variable definition: The verilog is used to define statement to associate a state value to each name (Armstrong, 1962). We simply transformed the data directly from the state diagram. The state name and the state value are inside the circle that define each state as:

```
//
// Define output state in terms of input variables
//                - - - - - - - - - - - output (subtract)
//                + - - - - - - - - output (swap)
//                1 + - - - - - - - output (load XR)
//                1  1 - - - - - - - output (select XY)
// extract bit    1  1  1 - - - - - - output (load YR)
//                1  1  1  1
                  V  V  V  V
‘define S1        d  d  d  d      // idle state, all bit set to d
,define S2        0  0  0  0      // extract bit == 0 here
‘define S3        1  1  1  1      // extract bit == 1 here
```

Output assignment section: Continuous assignment is used to assign an output to its associated state bit, this continuous assignment will synchronize to a zero delay wire i.e., the output name and the registered bit are synonyms. If there are asynchronous inputs to the state machine; this is a good stage to instantiate synchronizer (Kohavi, 2001; Barthel, 2004).

```
// Define state flip flops
   reg [5:0] state;
// continuous assignment of state bits to outputs
   Assign {output: substract, swap, load XR, select YR, load YR} = [5:0],
// synchronize asynchronous inputs if needed/occurred
```

Design method: This design utilized classical method of design using purely State Table, Excitation Table and K-Maps that describes the D flip flop characteristics equations. The following procedures/methods were taken into consideration.

- First the State Table was developed, which defines the control unit of the GCD processor
- The State Table was transformed into Excitation Table, where the inputs, present State and next state are clearly defined (Paul and Winfield, 1995)
- Logic Equations or functions were derived from the Excitation Table using K-Maps. However, in using the K-Map, the following sequence of rules for grouping 1's terms were carefully observed in any of the following manners:
    - First cover circles, all isolated 1's will be described by all n variable and require no further attention
    - Next, each remaining term is considered separately, if it can be group in more than one way. Another term is considered otherwise and is included in the largest possible rectangular coverage that is in a power of 2
    - Exhaustion application of the rule, is considered if at least one term remained, which is being grouped in more one way arbitrarily selected and return to rule 2
    - The simplified logic function becomes completed as soon as all terms are covered (Ronald and Neal, 2001; Muller, 1996)

Method I: The State Table that defines the control unit of the GCD processor is developed, where, S = 00, S1 = 01, S2 = 10 and S3 = 11 presenting digital codes in binary form.

$\underbrace{[(XR>0)\ \ (XR\geq YR)]}_{\downarrow}$

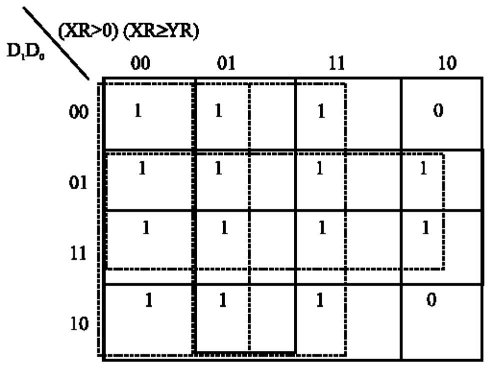
Inputs	0d	10	11	Subtract	Swap	Select XY	Load XR	Load YR
Begin	S3	S1	S2	0	0	1	1	1
Swap	S2	S2	S2	0	1	0	1	1
Subtract	S3	S1	S2	1	0	0	1	0
End	S3	S3	S3	0	0	0	0	0

Method II: The Excitation Table developed from the State Table in method II is as:

Inputs	Present state	Next state	Subtract	Swap	Select XY	Load XR	Load YR
(XR>0)(XR=YR)	D_1D_0	$D_1^+D_0^-$					
0d	0 0	1 1	0	0	1	1	1
0d	0 1	1 0	0	1	0	1	1
0d	1 0	1 1	1	0	0	1	0
0d	1 1	1 1	0	0	0	0	0
10	0 1	0 1	0	0	1	1	1
10	1 0	1 0	0	1	0	1	1
10	0 1	0 1	1	0	0	1	0
10	1 1	1 1	0	0	0	0	0
11	0 0	1 0	0	0	1	1	1
11	0 1	1 0	0	1	0	1	1
11	1 0	1 0	1	0	0	1	0
11	1 1	1 1	0	0	0	0	0

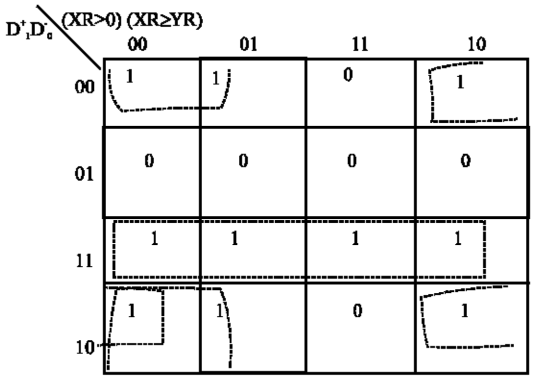
Method III: K-Map was employed to develop the Boolean expression of the Sum Of Product (SOP). As in case I:

Case I: Map for D_1^+



$$D_1^+ = \overline{(XR>0)}+(XR\geq YR)+D_0 \qquad (1)$$

Case II: Map for D_0^+



$$D_0^- = D_1D_0+(XR>0).\overline{D}_0+(XR\geq YR)\ (10)\ (4) \qquad (2)$$

Similarly the following were obtained in the same manner:

$$Subtract = D_1D_0 \qquad (3)$$

$$Swap = D_1D_0 \qquad (4)$$

$$Select\ XY = D_1D_0 \qquad (5)$$

$$Load\ XR = D_1+D_0 \qquad (6)$$

$$Load\ YR = D_1 \qquad (7)$$

To realize the GCD all the Boolean expression has converted to NANDs as follows (Almaini *et al.*, 1980; Lee and Davidson, 2005; Dietmeyer, 1999):

$$\overline{\overline{D^+_1}} = \overline{\overline{(XR>0)+(XR\geq YR)+D_0}} \qquad (8)$$

$$D^+_1 = (XR>0)+(XR\ \ YR)\,D_0$$

$$\overline{\overline{D^-_0}} = \overline{\overline{D_1D_0+(XR>0).\,\overline{D_0}+(XR\geq YR)}} \qquad (9)$$

$$\overline{D^-_0} = \overline{\overline{D_1D_0}\,\overline{(XR>0).\overline{D_0}}(XR\geq YR)}$$

$$\overline{Subtract} = \overline{D_1\overline{D}_0} \qquad (10)$$

$$\overline{Swap} = \overline{\overline{D}_1D_0} \qquad (11)$$

$$\overline{Select\ XY} = \overline{\overline{D}_1D_0} \qquad (12)$$

$$\overline{Load\ XR} = \overline{\overline{D_1}+\overline{D_0}} = \overline{D_1D_0} \qquad (13)$$

$$\overline{Load\ YR} = \overline{D_1} \qquad (14)$$

The logic circuit of Mealy digital machine was realized using the Boolean expression from (8-14) as depicted in Fig. 3.

The following hardware language was used to instruct the machine to compute the GCD of (40, 24) (Shahdad, 1986; Shiva, 2005).

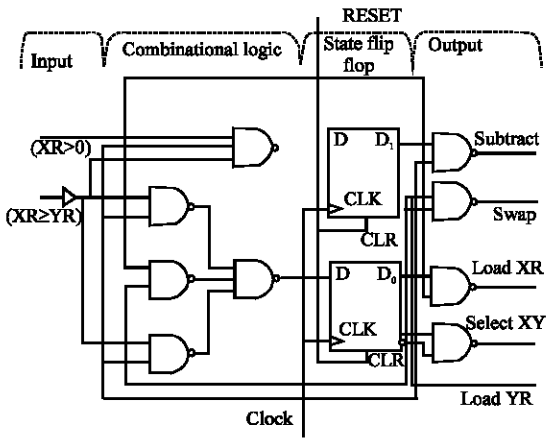


Fig. 3: Computed Mealy digital state machine

```
gcd (in X, Y; out: Z)
reg XR, YR, TEMPR;
XR: = X;
YR = Y;                    Input data
While, XR>0                Do begin
TEMPR: = YR;               Then begin (Swap XR and YR)
YR: = XR                   End
XR: = XR-YR                Subtract YR from XR
End
Z: = YR                    Output result
End gcd
```

Condition	Action	
XR>0: XR>YR	XR = 40; YR = 24;	
XR>0: XR≤YR	XR: = XR − YR = 16;	
XR>0: XR<YR	YR = 16; XR = 24;	XR: = XR − YR = 8;
XR>0	YR: = 8; XR = 16;	XR: = XR − YR = 8;
XR≤0:	YR = 8; XR = 8;	XR = XR − YR = 0
	Z = 8	
	GCD (40, 24) = 8	

REFERENCES

Almaini, A.E.A., M.A. Moosa and N.M. Aziz, 1980. Computer aided design of groups of exclusive logic function. Digital Proces., 6: 227-243.

Armstrong, D.B., 1962. A programmed algorithm for assigning internal codes to sequential machines. IRE Trans. Elect. Comput., 11: 446-472.

Barthel, D., 2004. A remark concerning minimization of expression of boolean algebra in the case of don't care condition. Ekektron Inf. Verarb Kyben Germany, 19: 393-396.

Dietmeyer, D.L., 1999. Logic design of fan, in limited NAND circuits. IEEE Trans. Comput., 18: 11-22.

Hallbaueyer, G., 1998. Procedures of state reduction and assignment in one step in synthesis of asynchronous sequential circuit. Proceeding of the International IFAC Symposium on Discrete Systems, (ISDS'98), Riga, Latvia, pp: 272-283.

Hill, F.T. and G.R. Peterson, 1997. Introduction to Switching Theory and Digital Design. 2nd Edn., John Wiley and Sons, New York, pp: 986-989.

Kohavi, Z., 2001. Secondary state assignment for sequential machine. IEEE Trans. Elect. Comput., 13: 193-203.

Lee, H.P. and E.S. Davidson, 2005. A transformation for NAND network design. IEEE Trans. Comput., 21: 12-20.

Muller, D.E., 1996. Application of boolean algebra to switching circuit design and error detection. IRE Trans., 3: 6-12.

Papadinatou, C.A. and D. Sarma, 1972. An approach to sequential circuits construction in LPI programmable array. IEEE Proc., 130: 159-164.

Paul, H. and H. Winfield, 1995. The Art If Electronics. Cambridge University Press, Cambridge, pp: 422-448.

Ronald, T.J. and W. Neal, 2001. Digital System, Principles and Application. 8th Edn., Pearson Education Pte. Ltd., USA., pp: 203-213.

Shahdad, M., 1986. An overview of HDL language and technology. Proceedings of the 23rd ACM/IEEE Design Automation Conference, (DAC'86), Piscataway, New Jersey, USA., pp: 320-326.

Shiva, S.G., 1980. Combinational logic synthesis from HDL description. Proceedings of the 17th Design Automation Conference, June 23-25, Minneapolis, Minnesota, United States, pp: 550-555.