SELF-CLOCKED SEQUENTIAL CIRCUITS: - A DESIGN EXAMPLE

F. Aghdasi and A. Bhasin
Department of Electrical Engineering,
University of Botswana, Gaborone
email: aghdasi@mopipi.ub.bw

Asynchronous sequential circuits offer improved speed of operation when compared to their synchronous counterparts. However, the standard methods of asynchronous design require careful examination of the flow table for possible critical races and hazards. This complicates the design procedure and often leads to extra states and additional hardware. A number of new design methodologies which involve locally generating a clock and using it to self synchronize the machine have been proposed. Such clock signals are generated whenever an input changes, or by controlled excitation whenever a change of inputs necessitates a change of state. All such designs, where the circuit is timed by locally generated clocks, are called Self-Clocked Sequential Circuits. This paper uses a design methodology for the State variable toggling through data driven clocks to implement a Direct Memory Access Controller (DMAC) as a design example. The design is simulated on software and also implemented using discrete hardware components. The methodology can be extended to parallel controllers for neural networks and automated using state assignment techniques already developed for synchronous parallel controllers.

Keywords: Sequential circuits, synchronous, asynchronous, neural networks

1 INTRODUCTION

Digital circuits are normally designed as networks of interconnected components. There are two approaches to the coordination of such components: Synchronous and Asynchronous

1. The components in a Synchronous Circuit operate under the control of a common clock. All the operations must take place at the correct time within each clock cycle. The speed of operation of synchronous circuits is always limited by the slowest element.
2. Asynchronous means without synchronization and in sequential machines it means without a synchronizing clock. **Rather, these machines respond directly to input changes and do not have to wait for a clock edge**. The basic reason for asynchronous designs is the enhanced speed of operation. Asynchronous controllers respond immediately to input changes and are much faster than their synchronous counterparts.

Recently there has been a great resurgence of interest in asynchronous circuits [1]. This interest stems partly from an increase in (asynchronous) communication activity in digital circuits and partly from a desire to achieve higher performance with lower energy consumption and lower design cost. As the integration density of very large scale integration (VLSI) circuits increases, synchronous circuits face clock skew and power distribution problems. The purpose of this paper is to introduce a self-clocked methodology that combines the advantages of asynchronous and synchronous designs.

1.1 Advantages of Asynchronous Sequential Circuits -

Much of today's synchronous logic design is based on two major assumptions [2]: all signals are binary and time is discrete. Both of these assumptions are made in order to simplify logic design. The assumption that signals are binary permits us to use Boolean algebra to describe and manipulate logic circuits. The assumption that time is discrete permits us to ignore hazards to a large extent.

Asynchronous digital circuits keep the assumption that signals are binary but remove the assumption that time is discrete. This has the following potential benefits:

(a) No Clock Skew: - Clock skew is the difference in arrival times of the clock signal at different parts of the global circuit. As clocks get faster, and VSLI chip density increases, tracks get thinner and it becomes increasingly harder to keep clock skew within tolerable limits. Since asynchronous circuits, by definition, have no global clock, there is no need to worry about clock skew. In contrast, the designer of a synchronous circuit must often slow down its operation in order to accommodate the skew.

(b) Power Efficiency: - Asynchronous circuits have the potential for very low power consumption since each module operates only when there is data to process hence saving power, which would have been consumed by modules clocked by a global clock in synchronous circuits while there is no data to process. This is an increasingly important issue for VSLI, especially since more and more systems are becoming portable. For these portable systems the advantage of lower power consumption and simpler power distribution is self-evident: longer battery life.

(c) Easing of Global Timing Issues: - In circuits such as synchronous microprocessors, the clock rate, and thus performance, is dictated by the slowest (critical) path. Thus, the design must be carefully optimized to achieve the highest clock rate: this optimization must be applied to rarely used portions of the circuit. Since many asynchronous circuits operate at the speed of the path currently in operation, rarely used portions of the circuit can be left unoptimized without adversely affecting overall performance.

(d) Better Technology Migration Potential: - Circuit functions are often implemented in several different technologies during their lifetime. Greater performance for synchronous circuits can often be achieved by migrating all system components to a new technology, since the overall performance is based on the slowest (longest) path. In many asynchronous circuits, migration of one or more critical components i.e. upgrading or replacing individual modules, can improve average performance since system performance depends only on the current active path.

(e) Modularity: - Different synchronous designs may have different clock requirements, and hence merging two components into a common system may prove very difficult. The modular approach to asynchronous circuits where each modular part making the whole system is self-timed avoids this problem.

(f) Automatic Adaptation to Physical Properties: - The delay through a circuit can vary with physical variations such as temperature, fabrication and power supply. Synchronous circuits must clock the system according to the worst possible combination of factors. Asynchronous systems on the other hand always run at the maximum possible rate under the current operating conditions as long as the design can guarantee the integrity of the signals.

1.2 Problems Associated with Asynchronous Circuits: -

Even though Asynchronous controllers may be advantageous over the synchronous ones in terms of speed, they have a few drawbacks. These include problems such as races, hazards, and the use of delay elements. Adequate consideration must be given to these problems during the design stages to avoid system malfunctions due to races and hazards.

(a) Races: - For an Asynchronous circuit to be of use, it must have more than one external input. If two of the multiple inputs are meant to change at the same time (simultaneously), this can never be guaranteed for a real circuit and one of them will always change slightly before the other. This is known as a ***Race Condition***, i.e. whenever the next state differs in more than one variable from the present state. This creates a possibility of unequal propagation delays within the circuit and may cause the circuit to reach a state other than that intended. This is called a ***Critical Race***. All other races are non-critical.

(b) Hazards - A combinational circuit is said to contain a ***Hazard*** if, in response to one or more inputs, an output momentarily assumes an incorrect value. Hazards normally occur in combinational logic circuit implementations of Boolean functions but do not prevent these circuits from performing their specified functions. A critical race hazard is one which subsequently causes the system to enter unwanted states (and operate incorrectly). Asynchronous sequential circuit design must not generate critical race hazards.

(c) Delay Elements: - All physical elements have some propagation delay associated with them which can generally be assigned upper bounds. These delays play an important part in determining the maximum allowable clock frequency in synchronous circuits. Delay elements are deliberately inserted into the asynchronous system by the designer. These are added in the path of the clock to ensure that the data is stable before the next transition occurs and suggests that the designer has to make extra circuitry to accommodate the delay elements. This complicates the overall circuitry of the system.

(d) To avoid the use of delay elements in self clocked designs, it is possible to design templates that ensure a longer delay for the path of the clock than the data. The Use of T type flip-flops as memory elements in the self-clocked schemes is one effective method to avoid delay elements. By generating data driven clocks and clocking only those registers that need to change, the need to establish stable data before the arrival of the clock edge is removed.

2 SELF CLOCKED DESIGN METHODOLOGY: - STATE VARIABLE TOGGLING THROUGH DATA DRIVEN CLOCKS

This method [3] proposes using a set of combinational logic blocks and a pair of master slave latches for each state variable. Each master-slave flip-flop is connected in the toggling mode such that every input clock pulse complements the output. **There are no clock pulses unless the established values of the inputs and the present state require a change in the next state values, and even then, the master latches of those next state bits which do not need to change receive no pulses**. The notable feature of this arrangement is that each variable is being clocked separately. The clock logic of each state variable generates a pulse only when that particular state variable needs to be toggled. The simplicity of this method enables it to be implemented directly from the flow table or an Algorithmic State Machine (ASM) Chart.

The basic design methodology can be summarized as follows.

- Construct the ASM Chart for the Asynchronous controller
- Allocate a separate flip-flop per state

- Establish the initial condition of the flip-flop
- Identify all the conditional paths that require each path to be active
- Identify the condition that requires the machine to leave that state
- Generate a clock function for each flip-flop to set and reset it

The ASM chart represents the machine and emphasizes the creation of a clear and detailed definition of the algorithm prior to implementation in detailed hardware design. It consists of a network of states with unconditional outputs, branches and conditional outputs.

The self clocked methodology uses one flip-flop per state. It is clear that at any one time only the flip-flop corresponding to the active state in the ASM chart is active. The condition that activates the next flip-flop also deactivates the previous one. The system reset puts the machine in the initial condition, normally the idle state. Thereafter the toggling flip-flops carry the machine through the various states dictated by the ASM chart.

3 DIRECT MEMORY ACCESS CONTROLLER (DMAC)

As a design example, a Direct Memory Access Controller (DMAC) has been implemented using the abovementioned methodology. **The example only serves to show how this design methodology can be used in practical designs. However, Direct Memory Access (DMA) controllers, in reality are very complex devices, and this example in no way provides an alternative asynchronous device for the present clocked controllers available in the market today.** The various input and output signals based of the 82C37A LSI and Z8410 DMA controllers [4 & 5] were studied to come up with an asynchronous circuit that gives similar signals. There are many internal registers in these controllers that we have not concerned ourselves with.

3.1 What does a DMAC do?

Direct Memory Access (DMA) [6], is an input/output strategy that moves data between a peripheral and the CPU's main memory without the direct intervention of the CPU itself. DMA provides the fastest possible means of transferring data between an interface and memory, as it carries no CPU overhead and leaves the CPU free to do useful work.

The DMA technique provides a direct access to the memory while the processor is temporarily disabled. This allows data to be transferred between memory and the I/O device at a rate that is limited only by the speed of the memory/peripheral devices in the system or the DMA controller. The DMA controller issues signals to the peripheral device and main memory to execute read and write commands. It takes over control of the address and data buses for the duration of the data transfer and then returns control of these buses on completion of the data

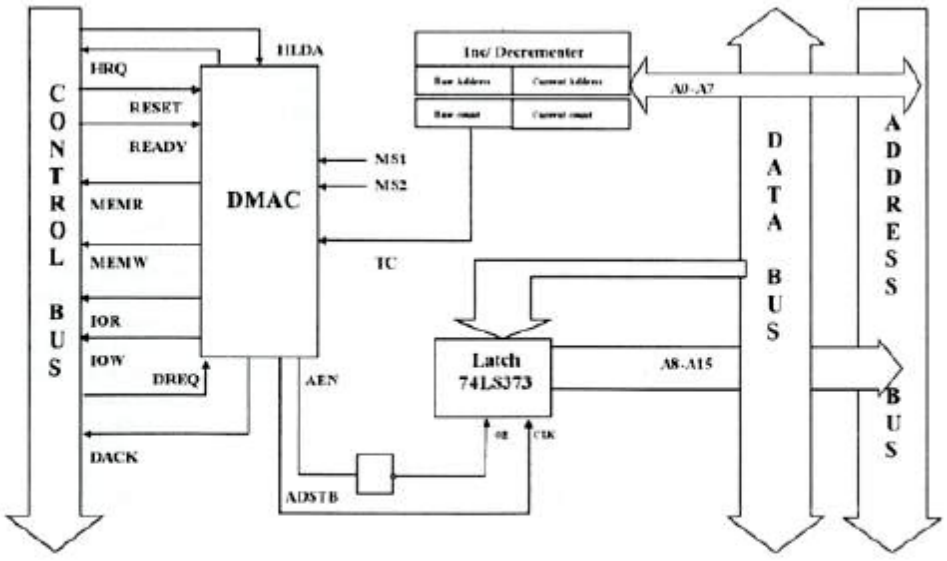


Figure 1. DMAC Block Diagram

transfer.

3.2 Asynchronous DMAC Design

From studying the various synchronous DMA controllers currently in use, the following block diagram (Figure 1) can be drawn. For the

INPUTS	OUTPUTS
X1: DREQ DMA request input used by peripheral devices to request DMA service.	Z1: HRQ Hold request output to the microprocessor to request control of the system bus
X2: HLDA Hold Acknowledge, active high input from the microprocessor indicating that it has relinquished control of the system buses.	Z2: DACK DMA acknowledge used to acknowledge a DMA request
X3: MS1 Mode select pin 1, used to specify the type of data transfer	Z3: AEN Address enable used to enable the DMA address latch and disable any buffers connected to the microprocessor
X4: MS2 Mode select pin 2, used to specify the type of data transfer	Z4: ASTB Address strobe used by the DMA controller to latch address A8-A15 during DMA transfer.
X5: Read/Write ready. A signal to the controller by a peripheral when it is ready for a read or write operation.	Z5: MEMR Memory read signal used to access data from the selected memory location during a DMA read or a memory-to-memory transfer.
X6: TC Terminal count signal, originates from the increment/decrement register to indicate data transfer completion in the block transfer mode	Z6: MEMW Memory write signal used to write data to the selected memory location during a DMA write or a memory-to-memory transfer.
RESET: An active high input that clears all the temporary address registers (base and current) and brings the device in the idle state (T1)	Z7: IOR I/O read signal used to access data from a peripheral during a DMA write transfer.
	Z8: IOW I/O write signal used to load data to a peripheral during a DMA read transfer

Asynchronous controller, the design can be reduced to a total of seven input signals and eight output signals. For ease of understanding, symbols have been assigned to the various input and output signals. A description of each follows.

DMA TRANSFERS: - *Simultaneous* DMA provides fastest transfers, with read and write operations performed at the same time. This requires MEMR and IOW (or IOR and MEMR) to be active simultaneously. In this way, data does not flow through the DMA Controller, but rather moves directly from the memory to the I/O port (or vice versa) [7].

Two types of data transfers are considered for the design example

i. Memory to Peripheral
ii. Peripheral to Memory

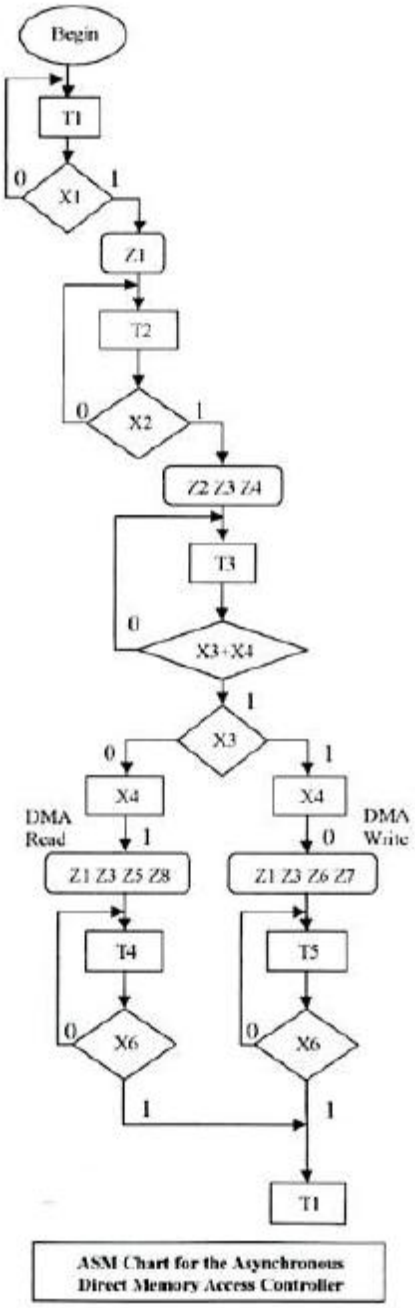
A DMA Read transfers data from the memory device to the I/O device while a DMA Write transfers data from an I/O device to memory. In both operations, the memory and I/O devices are controlled simultaneously and that is why the system contains separate memory and control signals.

The DMAC works with two address registers called the base address register and the current address register. These are synchronized with the main system clock and the memory and peripheral devices. The base address holds the starting address for the DMA operation and the current address register contains the address of the next storage location to be accessed. Writing a value into the base address register automatically loads the same value into the current address register. In this way the current address register points to the starting memory or I/O address. These registers are in fact internal registers of the 82C37A DMA controller but for the asynchronous design we are only concerned with the TC input signal that they provide and not with the actual workings of the registers.

Additionally, the DMA controllers that were studied have two word count registers called the base count register and current count register. The number of bytes that are to be transferred are specified by the value in the base count, and the current word count register tells how many bytes remain to be transferred. For the design example, how these values are updated is not a concern and only the TC signal that becomes active indicating transfer completion is used.

HANDSHAKING: - Initially at start the controller is in state T1 (initial state). If any error occurs in the DMA transfer process a reset signal from the

microprocessor brings the controller back to state T1. When a peripheral wants to perform a DMA operation it makes a request to the controller at its DREQ (X1) input by switching the input to logic 1. In response to an active DMA request, the DMA controller switches the Hold Request (Z1) output to logic 1 (State T2). This signals to the microprocessor that the DMA controller needs to take control of the system bus (Refer to ASM Chart)



ASM Chart for the Asynchronous Direct Memory Access Controller

The microprocessor responds within a few clocks, by suspending the execution of the program and placing its address, data and control buses in their high impedance states and signals this to the DMA controller by switching the Hold Acknowledge (X2) input of the controller to logic 1. The high impedance state causes the microprocessor to appear as if it has been removed from its socket and allows external memory or peripheral devices access to the system buses (Refer to ASM Chart)

When the DMA Controller has control of the system buses, it tells the requesting peripheral device that it is ready by giving out a DMA acknowledge (Z2) signal (State T3). The DMA Controller also outputs the high-order memory address on D_0-D- and raises AEN (Z3) and ADSTB (Z4). The address represents the source or destination of the data transfer. AEN enables the tri-state outputs of the latch. This completes the Peripheral-Controller-Microprocessor request/acknowledge handshaking, giving the peripheral device direct access to the system buses and memory under control of the DMA Controller.

DMA READ - A DMA Read transfers data from the Memory device to the I/O device i.e. Memory Read and I/O write. After the Handshaking procedure is completed the controller is in state T3 and depending on the Mode select pins (MS1 and MS2) the controller can now take one of two paths, namely DMA Read or DMA Write.

MS1	MS2	
0	0	Do Nothing
0	1	DMA Read
1	0	DMA Write
1	1	Do Nothing (For Future improvements)

For the DMA Read Mode input signal X3 is 0 while X4 is 1. The controller now enters state T4. The HRQ (Z1), AEN (Z3), MEMR (Z5) and IOW (Z8) outputs are turned ON. The active MEMR and IOW signals enable data transfer from memory to the I/O Device while the HRQ signal is used to hold the buses and AEN enables the respective addresses. The DMA Read continues in state T4 until the Terminal Count (X6) signal is received, indicating that the transfer is over. The controller then relinquishes control of the system buses and returns to State T1. (Refer to ASM Chart)

DMA WRITE - The DMA Write Mode is enabled when signal X3 is 1 while X4 is 0. The controller now enters state T5 and the HRQ (Z1), AEN (Z3), MEMW (Z6) and I/OR (Z7) outputs are turned ON. The active MEMW and IOR signals enable data transfer from the I/O Device to memory while the HRQ signal is used to hold the buses and AEN enables the respective addresses. Similarly to the DMA Read, the DMA Write transfer occurs in state T5 until the Terminal Count (X6) signal is received, indicating that the transfer is over. The controller

then relinquishes control of the system buses and returns to State T1. (Refer to ASM Chart)

From the design methodology described in chapter two, each state in the ASM chart is assigned a separate Flip-flop. This means that, since the Controller has five states, five D-type flip-flops are needed for implementing the controller.

HARDWARE IMPLEMENTATION: -The design example has been implemented using discrete hardware components. These include 74LS74 (D Flip flops), 74LS04 (NOT gates), 74LS08 (two input AND gates), 74LS32 (two input OR gates), toggle switches, 5V power supply and various colour LED's.

INITIAL CONDITIONS OF THE FLIP-FLOPS: - Initially state T1 is ON while states T2-T5 are OFF. In order to obtain T1=1, the condition $\overline{CLR}$=1, $\overline{PRE}$ =0, must be established. This condition forces T1 to be ON irrespective of the clock and data inputs.

States T2-T5 are initially set to logic 0 by making $\overline{CLR}$=0 and later $\overline{CLR}$ is changed to logic 1. $\overline{CLR}$=1 is achieved by observing from the ASM chart which conditions clears a particular state through grounding of the input. The D inputs of T2-T5 are always kept at logic 1 by applying 5V. If a Particular state has to be set or cleared this can be achieved by making $\overline{CLR}$=0 hence forcing that state to logic 0 irrespective of the logic 1 that is always maintained at the D input. Under normal operation the $\overline{CLR}$ and $\overline{PRE}$ are kept at logic 1, and data at the D input will only be transferred to the Q output when a clock pulse is generated.

IDENTIFICATION OF ALL CONDITIONAL PATHS THAT ACTIVATE EACH STATE: -The condition that requires each state to be active is derived directly from the ASM Chart. The condition for activating a flip-flop and hence the required state, is obtained from the condition specified in the directed line going into that corresponding state ANDed with the previous flip-flop state. If there is more than one directed line going into a state, all the conditions are ORed.

In order to activate T1, there are three paths that lead to that state. There are two paths from T4 and T5 and another from T1 itself (see ASM Chart).In order to generate the logic expression that activates T1, T4 is ANDed with X6, T5 is ANDed with X6 and T1 is ANDed with $\overline{X1}$.These three expressions are then ORed and the logic expression that activates T1 becomes T4X6+T5X6+T1$\overline{X1}$.

The logic expression is used to generate a clock circuit that activates T1. However, there is an expression that defines that the controller is not moving out of it present state. In this case this expression is T1$\overline{X1}$ and this term can be omitted when making the clock circuits for the states. This is based on the knowledge that when the $\overline{CLR}$ and $\overline{PRE}$ are made to logic 1 while the clock is at logic 0, the Q output of the flip-flop will not change. This means that the expression for the clock circuit of state T1 reduces to T4X6+T5X6 and the circuit is simplified.

The expressions for the other states are derived in the same manner as follows.

CLOCK CIRCUITS	
State T1	T4X6+T5X6
State T2	X1T1
State T3	X2T2X1
State T4	T3$\overline{X3}$X4
State T5	T3X3$\overline{X4}$

The clock circuits for the states were obtained from the logic expressions derived above. Whenever a particular state is required to be active, its clock circuit generates a clock pulse, which sets that particular state ON and hence each state can be activated separately and individually. The condition that activates one state resets/clears the previous state.

PRESET AND CLEAR CIRCUITS: -The inputs of the $\overline{CLR}$ and the $\overline{PRE}$ terminals must be able to change between logic 1 and 0 as required.

State T1:-This state is cleared by X1 and the clear input is initially at logic 1. X1 is also initially at logic 0. In order to obtain $\overline{CLR}$=1 for the clear input, X1 is Inverted and ANDed. When X1=0 $\overline{CLR}$ will be 1. If X1 goes high the $\overline{CLR}$=0 and hence state 1 can be cleared. The clear expressions for states T2-T5 were worked our similarly. The preset inputs for these states were maintained at logic 1.

CLEAR CIRCUITS	
State T1	$\overline{X1}$
State T2	$\overline{X2}$ $\overline{RES}$
State T3	$\overline{X3}$ $\overline{X4}$ $\overline{RES}$
State T4	$\overline{X6}$ $\overline{RES}$
State T5	$\overline{X6}$ $\overline{RES}$

Preset State T1: - Initially $\overline{PRE}$ =0 for T1. At anytime when any of the states, T2-T5, become ON, the condition $\overline{PRE}$ =0 must be prevented because this condition forces T1 to be ON. The preset input must therefore be changed to logic 1. This is achieved by making sure that the state that goes ON is used to drive the preset logic of T1 to 0.

Combining the above results yields $\overline{PRE}$ T1=T2+T3+T4+T5+$\overline{RES}$

Output Expressions:- The expressions for the outputs Z1 to Z8 were worked out similarly to the set conditions for each state. The Tables below give a summary of the expression for the various inputs of the flip-flops.

State T1	
CLK T1	T4X6+T5X6
CLR T1	$\overline{X1}$
PRE T1	T2+T3+T4+T5+$\overline{RES}$
D Input	+5V

State T2	
CLK T2	X1T1
CLR T2	$\overline{X2}$ $\overline{RES}$
PRE T2	+5V
D Input	+5V

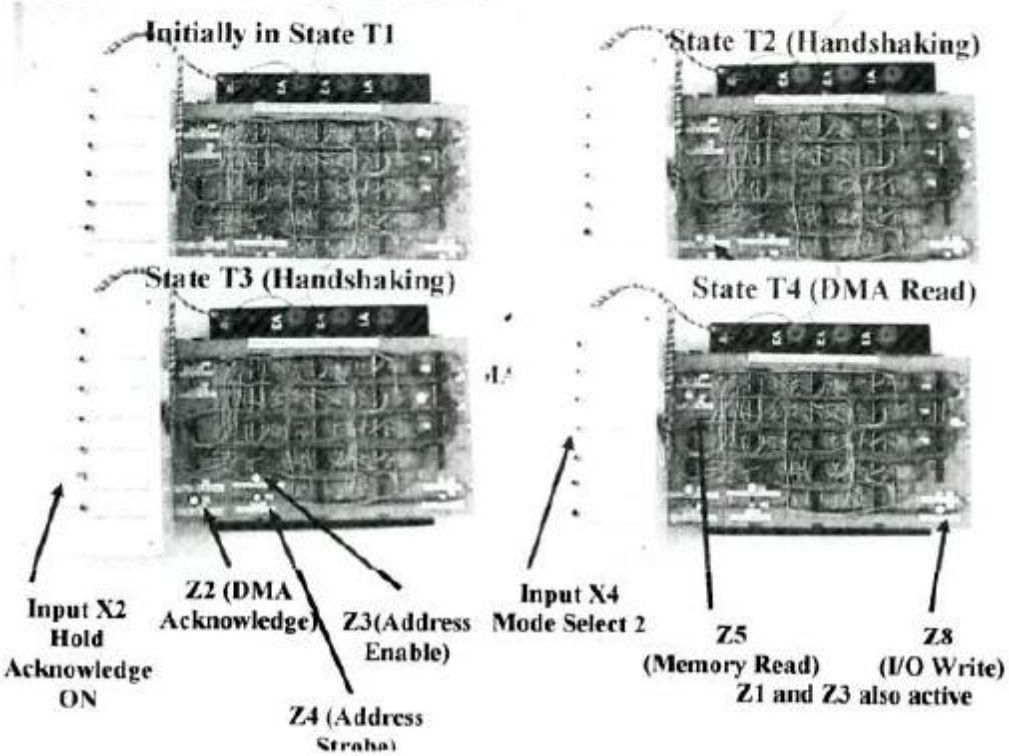
State T3	
CLK T3	X2T2
CLR T3	$\overline{X3}$ $\overline{X4}$ $\overline{RES}$
PRE T3	+5V
D Input	+5V

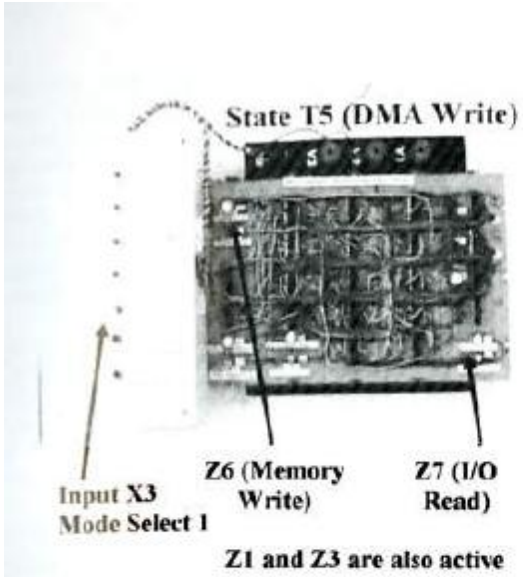
State T4	
CLK T4	T3$\overline{X3}$X4
CLR T4	$\overline{X6}$ $\overline{RES}$
PRE T4	+5V
D Input	+5V

State T5	
CLK T5	T3X3$\overline{X4}$
CLR T5	$\overline{X6}$ $\overline{RES}$
PRE T5	+5V
D Input	+5V

Outputs	
Z1	T2X1+T4X4$\overline{X3}$+T5X3$\overline{X4}$
Z2	T3X2
Z3	T3X2+ T4X4$\overline{X3}$+T5X3$\overline{X4}$
Z4	T3X2
Z5	T4X4$\overline{X3}$
Z6	T5X3$\overline{X4}$
Z7	T5X3$\overline{X4}$
Z8	T4X4$\overline{X3}$

In the circuit that was built, the signals were controlled using toggle switches. The circuit was simulated on Electronics Workbench and then built using the discrete hardware components described previously. Photographs of the circuit in its various states are given below.





4 RESULTS

The State variable toggling through data driven clocks methodology was used to develop an asynchronous Direct Memory Access Controller. The design was tested successfully using Electronic Workbench and also built using Discrete Hardware components. The controller responded immediately to input changes and only one state was activated at a time. Also, it moved from one state to another as dictated by the ASM chart and gave the required output signals.

5 CONCLUSIONS

Self clocked design methodology provides a systematic and easy to apply technique for designing asynchronous sequential circuits which do not need special consideration to avoid critical races and hazards. This method combines elegance, simplicity and reliability and can be implemented using standard components as well as semi-custom or VSLI technology [8].

The design example built in this project retains the fundamental characteristic of asynchronous circuits, that is, to respond immediately to input changes. The clock of each flip-flop is generated locally and separately whenever that particular flip-flop is required to change state. Therefore, the circuit is self clocked, but there is no synchronization of the flip-flops.

This technique uses more than the minimum number of flip-flops. However, this should not pose a great problem since Programmable Logic Devices (PLD'S) such as Logic Cell Arrays (LCA's) offer many flip-flops. Allocation of one flip-flop per state also means that the outputs associated with each state are readily available and no external decoding is required.

6 RECOMMENDATIONS AND FUTURE WORK

Asynchronous design offers the potential to minimize constraints imposed by traditional design. The methodology used here is only one of many interesting asynchronous design techniques that have been proposed recently.

The abundance of flip-flops in LCA's suggests that each state can be assigned its own flip-flop. The actual design method can be implemented directly from the ASM chart.

The self-clocking methodology can be applied to parallel controllers through decomposition, parallel state assignment or isomorphic encoding and of particular interest is its application in neural networks.

REFERENCES

1. Aghdasi, F., "Self-Clocked Asynchronous Controllers", University of Zimbabwe Publications, Harare, 1996, pp 34-58.
2. Brzozowski, J.A., "Asynchronous Circuits", Springer-Verlag, New York, 1995 pp 315-318.
3. Oommen, A., Murudkar, F., Aghdasi, F. "Self-Clocked Asynchronous Sequential Circuits", Botswana institution of Engineers-Conference Proceedings 1999 pp 166-175.
4. Uffenbeck, J., "Microcomputers and Microprocessors", Prentice Hall, USA, 1986.
5. Brey, B. "The Intel Processors", Prentice Hall, USA, 1986.
6. Chirlian, P.M., "Analysis and Design of Integrated Electronic Circuits", Harper & Row, New York, 1981.
7. Clement A. "The Principle of Computer Hardware", 2nd Edition, Oxford University Press, 1996.
8. Aghdasi, F. "Self-Clocked Controllers - A Practical Example", Computer Systems Symposium, COMPSYS-94, Stellenbosch, S.A., 3-4 October 1994, pp. 67-74.