Finite State Engines

Michael Forbes
Jian Xiong
Chinmay Athanikar
Raghavendra Jayaram

Department of Electrical Engineering
University of Tulsa

This report was prepared in partial fulfillment of the requirements of
EE 7063 Computer Engineering, a graduate course
Department of Electrical Engineering
John H. Letcher, Professor of Computer Science

Introduction

The past dozen years have seen a revolution in the computer industry. Twelve years ago, a single chip processor running near 500 MHz was not only unimaginable, but also impossible. Twelve years of technological innovation, including a continual decrease in both size and operating voltages, have made the 500 MHz single chip processor a reality. Discussion of these two innovations is outside the scope of this paper, which focuses on the changes in processor design techniques that have made this improvement possible.

Finite State Machines As Building Blocks

Modern processors, cache controllers and most peripheral device controllers are created as extremely complex finite state machines. Simple finite state machines serve as building blocks to form more complex machines, which are combined to make complete processors. A problem exists in that unless steps are taken, these finite state machines have always been vulnerable to a problem called metastability failure, where an asynchronous input causes an invalid state, which in turn causes an invalid answer. This paper discusses three models of finite state machines, including how each is affected by metastability, and we suggest that it is primarily the ability to avoid metastability failure by using the third model that has allowed modern microprocessors to achieve speeds which were previously unimaginable.

To completely understand finite state machines, as well as metastability, it is first necessary to understand the building blocks of the machines themselves: namely, combinatorial logic and D-type latches. A review of these building blocks follows, with definitions of important parameters and examples.

Combinatorial Logic

Combinatorial logic consists of a set of logic gates that implements a function $\mathbf{Y}(t+\lambda) = f(\mathbf{X}(t))$, where **X** is a finite set of digital inputs and **Y** is a finite set of digital outputs. The number of lines in **X** and **Y** does not have to be the same. λ is known as the propagation delay, and varies with each circuit. The nature of the function itself actually does not matter for this discussion, because as long as λ is finite and has an upper bound, the function can be used to make a finite state machine. The following timing diagram* (Figure 1) shows the operating characteristics of combinatorial logic.

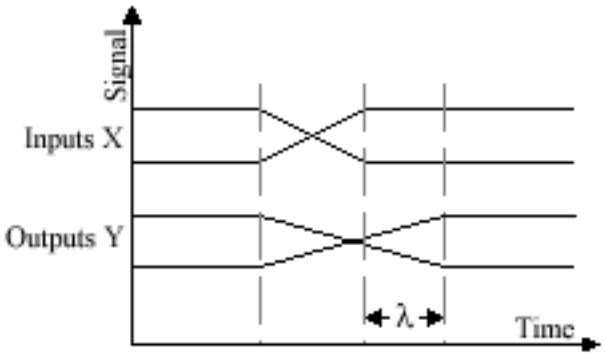

Figure 1: Timing characteristics of combinatorial logic

It is important to note a few things about this diagram. In this diagram, the outputs **Y** become unstable as soon as any of the inputs become unstable. While in an actual circuit there may be a delay between when an input becomes unstable and when an output goes unstable, it will be λ only for a very specific type of circuit, and in the general case will depend on which inputs are unstable. Therefore, the only safe engineering assumption is to assume that when **X** becomes unstable, so does **Y**. Similarly, when any signal in a set becomes unstable, we say that the entire set is unstable.

It is extremely useful to note that a standard memory device such as a PROM satisfies the definition of combinatorial logic. When the inputs (the address and some sort of select) are stable, the outputs become stable after a certain well-defined time. This means that we can use memory devices to implement the logic blocks of finite state machines, which will simplify immensely the task of designing the function block.

D Latches

The second building block of finite state machines is the D latch. The D latch provides a means of both holding values and guaranteeing stability. A D latch operates with a clock, and sets the values of the outputs **Q** to the values of the inputs **D** when an active clock is detected by a transition. When an active clock is not detected, the values of **Q** do not change, no matter what happens to **D**. The timing diagram for a rising edge-triggered D latch is shown below (Figure 2).

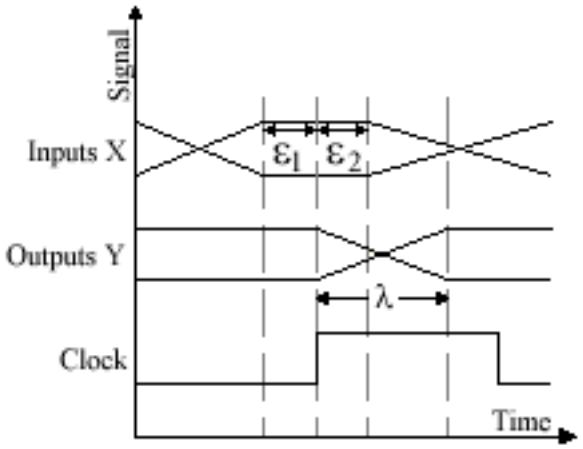

Figure 2: Timing characteristics of a D Latch

The three parameters that are germane to this discussion are ε_1, ε_2, and λ. In order for a D latch to operate properly, the inputs must be stable for a certain time before and after the clock pulse. These times are defined to be ε_1 and ε_2, respectively. The delay λ is defined as the time between the rising edge of the clock pulse and the instant when the outputs become stable. The inputs need only be stable for the region surrounding the clock pulse; the outputs will be stable for all time except during the period of instability

immediately following a clock pulse. The times, ε_1 and ε_2 are characteristics of a given latch which can be determined experimentally.

Metastability failure is a phenomenon that causes the previously given definition to fail. In order to understand the impact of metastability on circuit design, three different types of finite state machines will be discussed using the building blocks already defined. Therefore, we shall have abstracted the machines into moieties that can be used in future engineering calculations of performance and stability. These three machines all use D latches and function blocks, and all three can implement the same function, but all three have different characteristics, particularly with regard to metastability and with regard to the ability to predict stability and certain performance characteristics.

Definition of State Machine Models

We know a state machine, also called a sequential machine, is a system that can be described in terms of a set of states that the system may enter. A large majority of practical state machines use clocked flip-flops as the storage elements. The code that defines each state then corresponds directly to the code contained by the flip-flops.
There are three fundamental models for sequential machines or state machines. The following section will introduce these three models, including the Mealy machine, the Moore machine and the Letcher machine.

The Mealy Machine[1]
The general Mealy machine is shown in Figure 3. The input logic and output logic sections are made up of combinatorial logic blocks. The state register (flip-flop or latch) section contains the state of the system. A path is provided from state register output to the combinatorial logic block as a state feedback. Both input signals and present state signals drive the combinatorial logic to determine the next state of the system. The outputs are determined by the present state and the inputs of the machine. In the more general Mealy machine, the outputs depend on the present state and the present input signals. The outputs are functions of both the state and input signals. The outputs can change immediately after a change at the inputs, independent of the clock. A Mealy machine usually has asynchronous outputs.

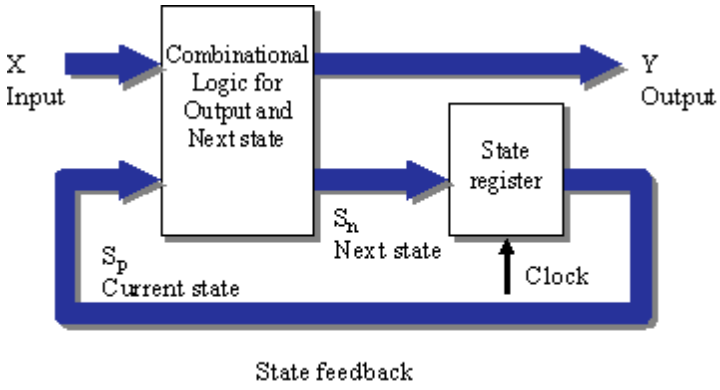

Figure 3: Mealy machine block diagram

It is easily seen from Figure 3 that the next state S_n, is a function of current state S_p, and the current inputs X. And the outputs Y is a function of the current state S_p and the current inputs X. In the other words, we may express Y and S_n as follows[1]

$S_n(t) = f (S_p(t), X(t))$ (1)

$Y(t) = h (S_p(t), X(t))$

Actually, the terms of the left had side of the above equation are valid only after finite propagation delays, a characteristic of the circuits.

The Moore Machine[1]

A slight variation of the Mealy machine is the Moore machine, which uses only the state register (flip-flops or latch) to drive the output logic described in the block diagram in Figure 4[1]. The state register provides the current state to the combinatorial logic block as a state feedback. And the combinatorial logic block maps the inputs and the current state into the state register to store the appropriate next state. In this case, the outputs are computed by a combinatorial logic for output block whose inputs are the state register's outputs. We can see the outputs are determined by the current state. So the outputs depend only on the current state. The outputs change synchronously with the state transition and the clock. This gives Moore machines an advantage in terms of disciplined timing methodology.

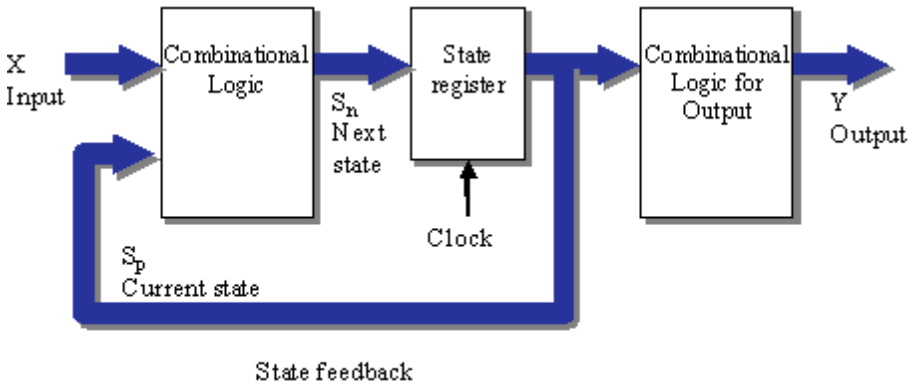

Figure 4: Moore machine block diagram

Like the Mealy machine, from Figure 4. we can see that the next state S_n is a function of current state S_p, and the current inputs X, but the outputs Y is only a function of the current state S_p. Here Y and S_n can be expressed as the follows[2]

$S_n(t) = f(S_p(t), X(t))$ (2)

$Y(t) = h(S_p(t))$

Again, the left had side terms in the above equation are valid only after preparation delays, characteristic of the circuits.

The Letcher Machine[4,5]

The Letcher machine is a relatively new type state machine. It is different from the other two state machines, the Mealy machine and the Moore machine. In general in the Mealy machine and the Moore machine, the basic architecture usually is Inputs→logic unit→ state register (such as flip-flops) →logic unit→outputs. The Letcher machine comprises three latches, such as a clock signal driven, edge triggered and parallel entry shift registers, and a function module (combinatorial logic block). The three latches are Input Latch, Output Latch and Feedback Latch, respectively. Its basic architecture is Inputs→ Latch→function module (combinatorial logic)→Latch → outputs. Figure 5[3] illustrates a typical Letcher machine. First, the Letcher machine uses latches to isolate or "freeze" a stable input on the rising edge of clock signal and produce and keep a stable output until the next rising edge of the clock signal. A path is provided from the Feedback Latch output to the combinatorial logic block as a state feedback. And both Input Latch's output signals and current state signals drive the combinatorial logic to determine the next state of the system. The current state and the output of Input Latch determine the input of the Output Latch. The input of Output Latch is isolated as a state value on the rising edge of clock signal and produces a stable output since the outputs, driven by Output Latch, change synchronously with the clock edge. The operation of the Letcher state machine is inherently synchronized by an outside supplied clock signal which can be altered upon demand. Here the synchronous output signals take effect at the next clocking event.

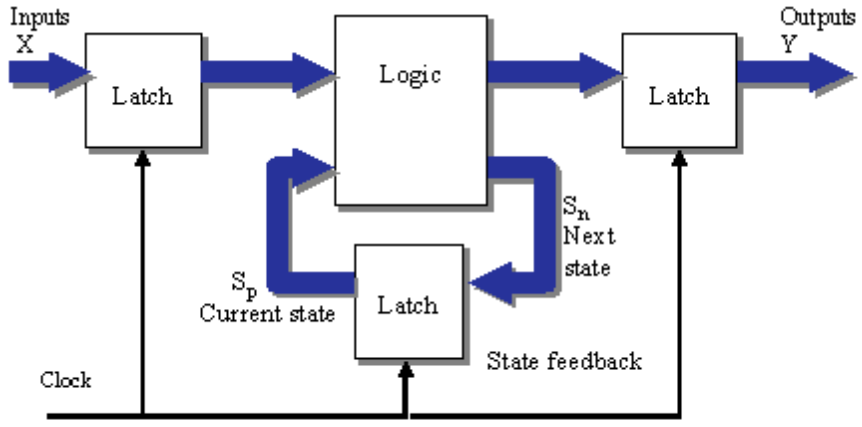

Figure 5: Letcher machine block diagram

A diagram of Figure 5 illustrates the outputs depend on the current state and the input signals. The outputs change synchronously with the outputs of the logic block and the clock edge. The next state S_n, is a function of current state S_p, and the current inputs X. And the outputs Y is a function of the current state S_p and the current inputs X. In the other words, we may express Y and S_n as follows.

$$S_n(t + \lambda) = f (S_p(t), X(t)) \quad (3)$$
$$Y(t +\lambda) = h (S_p(t), X(t))$$

Comparison of the Three Machine Models

Above we gave the description of three fundament models for the sequential machines. The three models have significant differences between them. Here these differences will be discussed.

The Mealy Model vs. The Moore Model

The two models of the sequential machines presented in the above section differ in the way in which the output signals are generated. We saw that the output depends only upon the state in the Moore machine. Therefore we always have an output available, except during the transition period when the machine is changing states. For a Mealy machine, however, its output depends on the current state and current input. So it can not produce the output unless an input signal is applied to the machine.

Another important different characteristic of these two model machines depends on whether the system is clock-driven or not. We refer to a variable as clock-driven if the variable changes value only at the time of a clock transition. Then, it is considered a synchronous variable. If a variable can change at a time not related to transitions of the reference clock, it is called an asynchronous variable. State machines are classified as synchronous or asynchronous. According to the general architecture of a Moore machine, the outputs are driven by the clock signal and change synchronously with the current state transition and the clock edge. Moore outputs are synchronous. In the Mealy machine, on the other hand, the output can change immediately after a change at the input, independent of the clock. So, Mealy outputs are asynchronous. The synchronous output characteristic gives Moore machines an advantage in terms of disciplined timing methodology. However, there is a synchronous variation of the Mealy machine, which we will describe later.

An interesting question that arises is whether we should use the Mealy or Moore model in the design of a sequential machine if we must use only these two models. Because it can associate outputs with transitions, a Mealy machine can often produce the same output sequence in fewer states than the equivalent Moore machine. Despite the Mealy machine's timing complexities, designers like the reduced state count. In general, fully synchronous state machines are much easier to implement and debug than asynchronous machines. If you were using discrete TTL components, you would usually prefer the Moore machine model, even though it may require more states. The asynchronous Mealy machines have the timing complexities, so they are difficult to design.

The Mealy and The Moore Models vs. The Letcher Model

In conventional state machines such as a Mealy machine or a Moore machine, the input signals are directly provided to a logic unit. We can see from Figure 4 that the Moore machine inputs are provided to the combinatorial logic, and the combinatorial logic produces output signals. We know ambiguous signals sometime might occur at the inputs of the combinatorial logic. During the transformation process used in the operation of the

state equation stores, there is a large period of time when the signals are considered unstable. This instability can cause jumbled, nonsensical data which are to be used as inputs. It can be seen that a metastable output can cause erroneous signals to be supplied to the state register and finally produces erroneous output signals. This is called as the metastability failure of a state machine. When the clock frequency is increased, the probability of Metastability failure will be increased. To make a Mealy or Moore machine run at high speed, we need to use high clock frequency. In this case, the machine will cause more erroneous signals at the output port.

The Letcher machine, however, is different from either the Mealy or Moore models. Its function module (such as combinatorial logic) is surrounded by "latches". The Letcher machine uses latches to isolate or "freeze" a stable input on the rising edge of the clock signal. The use of latches ensures that each input/output taken by a component is stable as long as stability conditions of the input latches are met. This model solves the problem of potential signal instability. The characteristic of the Letcher model allows it to be able to run at high clock frequencies.

The Letcher machine outputs are driven by the clock signal. Like the Moore model, the Letcher model is a synchronous state machine. The Letcher model exhibits a lot of advantages including the following:

1. The probability of circuit malfunction of Letcher machines due to *metastability failure* is identically zero as long as certain well defined timing conditions are met.
2. No hazards exist in Letcher machines.
3. The Letcher machine permits ultra high clock frequencies and still preserves absolute freedom from metastability failure.
4. The Letcher machine permits total freedom to slow clocks to any degree, so as to allow static operation.
5. It is easier to design large engines.
6. Unlike the other models, it is possible to calculate the maximum clock frequency of the Letcher machine, a definite advantage in using the machines inside other machines.

Static Operation

In the high speed state machine design, we usually hope this machine can run in a low power mode when it is not used. Since CMOS logic draws current only when it is switching, the power draw of a circuit is directly proportional to the clock frequency. The conventional state machine must be driven by a narrow range of frequencies. So it is desirable that a machine model permits total freedom to slow the clock to any degree. This is useful in the design of laptop computers to run in the low power mode while not being used. The ability to run as designed at essentially zero frequency is called static operation.

Edge-Triggered vs. Level-Triggered Latches

Clocked latches or flip-flops are designed to cause an output change either when the clock signal makes a transition or when this signal reaches some particular level. A given latch or flip-flop is referred to as an edge triggered device or as level-triggered device. These latches or flip-flops have been used in the state machine design.

Transitions (edges)

A clock signal has two types of transitions. These two types are classified as follows:

1) *Low-to-High transition* corresponds to a signal changing from low logic state to high logic state. When this type of transition permits change in the outputs, the latch or flip-flop is said to be *rising-edge triggered.*
2) *High-to-High transition* corresponds to a signal changing from high logic state to low logic state. When this type of transition permits change in the outputs, the latch or flip-flop is said to be *falling-edge triggered.*

Level-Triggered

Clocked latches or flip-flops cause an output change when this signal reaches some particular level. This type of latch or flip-flop is said to be *level triggered.*

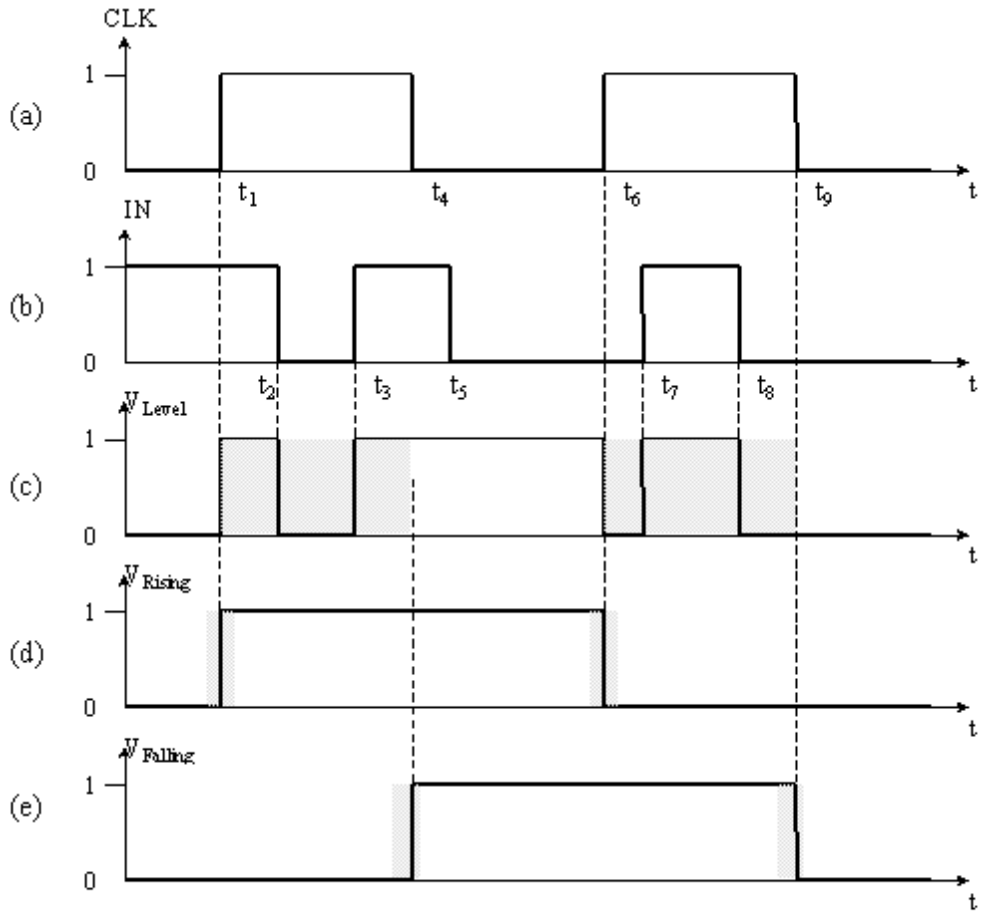

Figure 6: Time Waveforms showing differences between Level-triggered and Edge-triggered Latches
(a) clock signal, (b) Input signal, (c) Level-triggered latch output
(d) Rising-edge triggered flip-flop output, (e) Falling-edge triggered flip-flop output

Figure 6 shows the timing diagrams of level-triggered and edge-triggered latches or flip-flops. A periodic clock signal is shown in Figure 6a. Letting the input signal in Figure 6b be the input to the three different types of latches or flip-flops produces the output waveforms shown in Figures 6c, 6d, and 6e.

The Level-Triggered device's waveform is shown in Figure 6c. We can see the output signal changes with the input signal in (b) whenever the clock signal in (a) is high. At time t_1, t_2, t_3, t_7, and t_8 output (c) is seen to follow the input in this manner. When the clock signal goes low, the output signal (c) stores or "latches" the input signal value until the next time the clock signal goes high. This is also called a transparent latch.

The Rising-edge triggered device's waveform is shown in Figure 6 d. The output signal in Figure 6d is initially low. At time t_1, the clock signal in (a) experiences a rising edge and the output signal in (d) attains the logic value of the input signal in (b) and keeps it until next rising edge. At time t_6 (next rising edge), the output signal in (d) again attains

the logic value of the input signal. This output signal changes state only on the rising edge of the clock.

The Falling-edge triggered device's waveform is shown in Figure 6e. The output signal in (e) is seen to changes the logic state only when the clock signal goes the falling-edge of the clock signal and keep it until the next falling edge of the clock signal.

According to the differences between level-triggered and edge-triggered latches, we know level-triggered latch can change its output state whenever the clock signal is high. If level-triggered latches are used to construct a state machine, we will find that the state machine output can change very soon after a change at the input when the clock signal is high, independent of the clock. So level-triggered latches are usually used to design asynchronous state machines. On the other hand, edge-triggered flip-flops or latches are different from level-triggered latches as an edge-triggered flip-flop or latch can only change its output state when the clock signal experiences a falling-edge or rising-edge of its clock and keeps its output state until the next falling-edge or rising-edge of the clock signal. Its output is driven by the clock signal. So level-triggered latches are usually used to design synchronous state machines and also asynchronous state machines such the as Mealy machine.

Improvements To The Finite State Machine Models

Synchronous Mealy machine:

A conventional Mealy machine is asynchronous. Input changes lead to output changes, independent of the clock. This can play havoc when the outputs are signals that immediately control the data path. But because a Mealy state machine encodes control in fewer states, saving on the number of state registers, it is still desirable to utilize them. So it seems desirable to build a synchronous Mealy machine[4]. The simple way to construct a synchronous Mealy machine is to break the direct connection between inputs and outputs by inserting registers such as flip-flops. Figure 7 shows possible ways to construct a Mealy machine with synchronized outputs, using edge triggered devices: (a) only at the outputs, (b) at the inputs and outputs. We can see that the outputs are driven by the clock signal. the output are synchronous with respect to the clock.

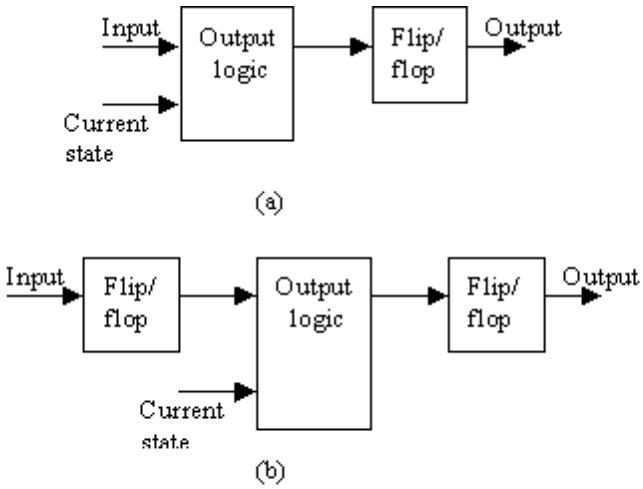

Figure 7: Block diagrams of Mealy machines with synchronous output

It can be seen that the synchronous Mealy machine is a Letcher.

Use a double-edge triggered latch unit

In the synchronous state machine design, the machine are used with a clock signal, and every time an effective edge of the clock signal occurs, the values on the outputs of the system are updated. The effective edge might be defined as the rising edge of the clock signal or the falling edge of the clock signal. Synchronous state machines are designed using any of a variety of different types of registers such as D flip-flops, JK flip-flops, latches and so on. The output of these registers is updated at the time that the rising edge or falling edge of the clock signal or by using level triggered devices. In synchronous machine design, the machine may be incorporated onto one or more integrated circuit chips with the clock signal provided from an external source along a printed circuit board trace. As the clock frequency is increased, however, it has become more difficult to provide such a high frequency clock signal to various chips. The traces will exhibit significant reactance at high frequencies. So the problem is magnified on printed circuit boards. Another factor is the frequency limitations of the technology. The high clock frequency can cause very high speed logic designs to be limited first by poor clock fidelity.

Figure 8 shows a diagram of double-edge triggered register/latch unit[5]. This latch unit is designed to update its output on both the rising edge and falling edge of a clock signal, so called double-edge triggered. In Figure 8[6], the double-edge triggered latch unit comprises a rising edge latch 1 and latch 2, parallel-connected, and multiplexer 3. Latch 2 has a clock input which is inverted to its enabling port. The two enabling inputs of latches 1 and 2 and the S select input of multiplexer 3 are connected to the system clock(CK). The latch 1 and 2 are rising edge-triggered latches in the system.

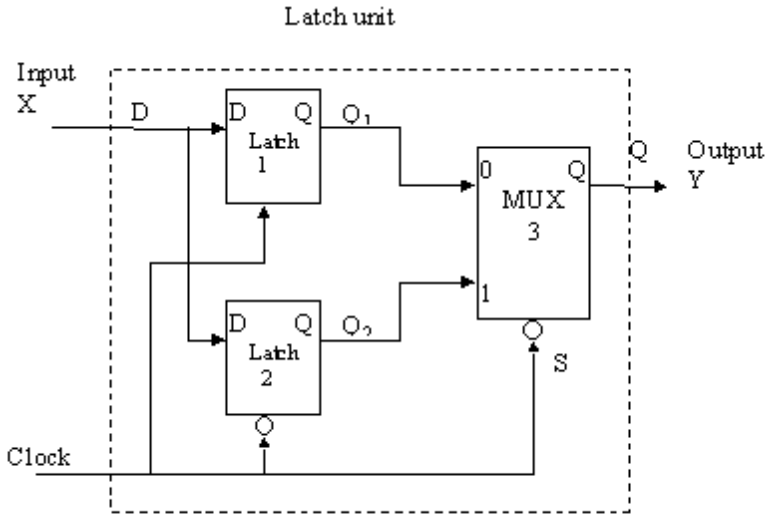

Figure 8: A block diagram of double edge triggered latch unit

Figure 9 is a timing diagram of a latch unit. When a rising edge occurs in the clock signal. The clock signal is connected directly to the clock of the latch 1, and input signal is connected directly to the input D of latch 1. Because latch 1 is rising edge-triggered, so the output of latch 1 will change from its previous output to a "new output" on the rising edge of the clock signal. The output of latch 1 will not change again (be frozen) until the next rising edge of the clock signal. The clock signal that is connected to the latch 2 is inverted. So this time a falling edge occurs in the clock signal of latch 2. The output of latch 2 will keep its previous value. In Figure 6, the select input S of multiplexer 3 is connected to the inverted clock signal. When the clock signal is high, the multiplexer 3 will select the output of the latch 1 as the output of the latch unit 1. This is shown on the timing diagram of Figure 7. When the next falling edge occurs in the clock signal, the output of latch 1 will keep its previous value. The output of latch 2 will change from its previous output to a "new output" on the rising edge of the clock signal. This output is frozen until the next falling edge of the clock signal. This time multiplexer 3 selects the output of the latch 2 as the output of the latch unit 1. It can be seen that the output of the latch unit 1 can change every half cycle of the clock signal.

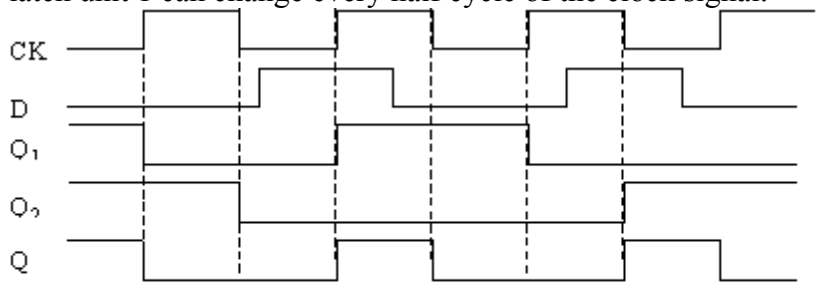

Figure 9: A timing diagram of a latch unit

In a synchronous state machine design, we can use the double-edge triggered latch unit to replace the conventional register, such as rising edge flip-flop or latch, in the state machine such as Moore machine and Letcher machine. In this case, the state machine can update its output and states on both the rising edge and falling edge of a clock signal. it can produce almost same output rate at the clock frequency to be one-half that of the clock frequency at the system which uses rising edge or falling edge flip-flops or latches.

For example, a typical synchronous state machine usually employs the rising edge triggered type of latch. A synchronous state machine can be designed using a double-edge triggered latch unit which updates the latch's output on both the rising and falling edges of a clock input. Figure 8 shows the timing diagrams of a typical synchronous state machine with rising edge triggered latches and a state machine with double-edge triggered latch unit. Comparison of these two timing diagrams in Figure 8 illustrates a synchronous state machine with a double-edge triggered latch unit that has one-half of the clock frequency of a typical state machine with rising edge latches, but has the same output rate. The use of double-edge triggered latch units in the design will make a synchronous state machine receive high er speed at a lower clock frequency.

Metastability

If either or both the setup or hold-time requirements for a particular clocked flip-flop are violated, the flip-flop may assume an undesirable state, which is referred to as a metastable state. The metastable state is characterized by a signal level at an output terminal of the flip-flop pausing at an amplitude intermediate between the logic one and logic zero level characteristic of an operating flip-flop. Rather than switching completely, the level falls back to the previous logic level. In a Metastable state, the intermediate signal amplitude level (between the high and low **threshold voltages) at the output** terminal of a flip-flop causes the output states of a functional building block to be unpredictable.[7]

Problems resulting from metastable operation of flip-flops are most frequently encountered when data or control signals are generated asynchronously with respect to the internal clock of the computer or other digital system. Such asynchronous signals can be generated in a variety of sources external to the computer, such as auxiliary equipment and including printers, other computers, and other such devices.

The failure of a metastable flip-flop to achieve a known logic state upon the active transition of the clock input signal could cause unacceptable errors in the operation of digital systems. In particular, the partial switching of the output level of a metastable flip-flop can produce an undesirable pulse of short duration called a "runt" pulse or "glitch", which may propagate partially or fully through downstream logic building blocks. In addition, an indeterminate delay in the return of the output signal level of a metastable flip-flop to its original state can cause system errors.

Quantitative Analysis of Metastability

The probability that the metastable state will occur can be given by the parameter, which is Mean Time between Failure (MTBF).

Lets take a simple example of a D latch and try to calculate the mean time between failure.[8]

For a typical, 1.5 micron CMOS D flip-flop, the MTBF can be calculated as

$$MTBF = 1/(A \cdot f_C \cdot f_D \cdot e^{-B \cdot Ts})$$

Where:
f_c is the Clock Frequency
f_d is the asynchronous data rate of the input.
T_s = Clock Period – Propagation Delay – Setup Time
And A and B are metastability constants that vary with construction and technology and system parameters

Example:

$$f_C = 20MHz, f_D = 5MHz, Tp = 30ns, T_{su} = 2ns, A = 4.0 \cdot 10^6$$

For nominal conditions (V=5.0V, Temp=25 C), B~3 * 10^9

$$MTBF = 1/\left((4 \cdot 10^6)(20 \cdot 10^6)(5 \cdot 10^6)\left(e^{-(3 \cdot 10^9)(18 \cdot 10^{-9})} \right) \right) = 7.08 \cdot 10^{14} s = 22.4 \cdot 10^6 \, years$$

For worst case (military) conditions (V=4.5V, temp=125C), B ~1.5* 10^9

$$MTBF = 1/\left((4 \cdot 10^6)(20 \cdot 10^6)(5 \cdot 10^6)\left(e^{-(1.5 \cdot 10^9)(18 \cdot 10^{-9})} \right) \right) = 1330s = 22.2 \text{ minutes}$$

With the same clock frequency but with different operating conditions, one can see that the MTBF values change drastically. This shows that we take a lot of risk of failure if we do not account for the possibility of metastability.

A large system consisting of many finite state engines, with the clocks presented to each finite state engines, can be viewed as a directed graph in the mathematical sense. The outputs of one engine can be presented as inputs to other. As long as the clocks to each finite state engine are adjusted so that the timing rules are followed, then within the system the MTBF (due to metastability) of the entire system is infinite.

Avoiding Metastability Using The Letcher Machine Model

The proposed state engine results in metastability failure free operation. As long as the functional block propagation delay never exceeds λ, the maximum delay, the Letcher State Machine cannot begin metastable operation. This means that by using a properly designed Letcher machine the problem of metastability failure can be avoided entirely. The further use of metastable resistant components will provide additional assurance that metastability is not a problem in Letcher State Machines.

Metastable Proof Flip-flop Designs

Recently progress has been made in the areas of metastable proof flip-flops[9]. Several examples, as well as their applications to finite state machine design, are discussed in this section.

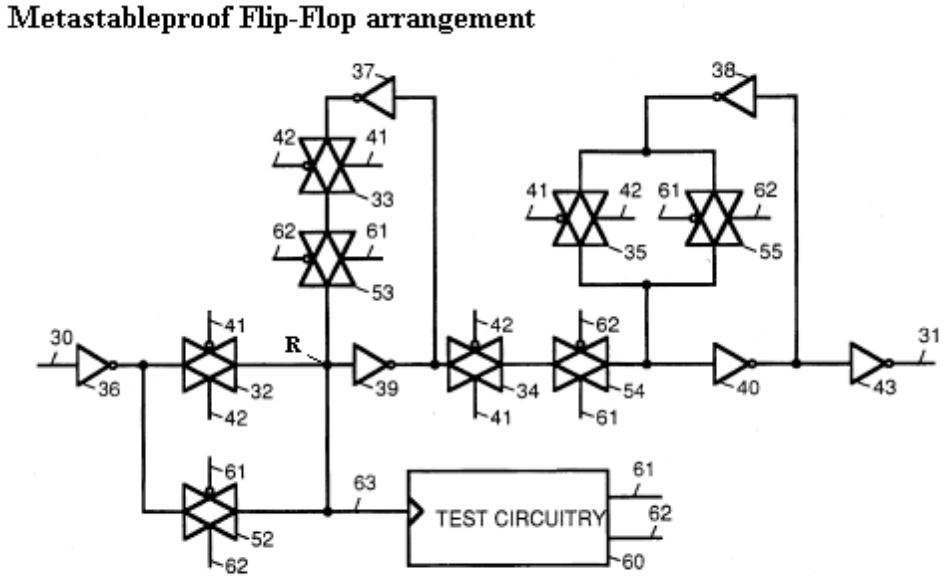

Figure 10: Metastable Proof Flip-flop arrangement

Fig. 10 shows a CMOS implementation of a metastable proof flip-flop. A flip-flop is used to capture a data value placed on D input 30. A Q output 31 holds a flip-flop output value. A clock signal (Clk) 41 and an inverted clock signal (|Clk) 42 are used to clock a pass gate 32, a pass gate 33, a pass gate 34 and a pass gate 35 within the flip-flop. The flip-flop additionally includes an inverter 36, an inverter 37, an inverter 38, an inverter 39, an inverter 40 and an inverter 43 , connected as shown.

Test circuitry 60 within the metastable proof flip-flop generates a data valid signal on a line 61 and an inverted data valid signal on a line 62. As long as the voltage of the signal on a node 63 is invalid (e.g., between 1.5 volts and 3.5 volts) the valid signal on line 61 is disserted that is driven low. And the inverted data valid signal on line 62 is asserted (e.g., driven high). When the voltage of the signal on node 63 is valid (e.g., less than 1.5 volts

or greater than 3.5 volts), the valid signal on line 61 is asserted (e.g., driven high). And the inverted data valid signal on line 62 is disserted (e.g., driven low).

The valid signal on line 61 and the inverted data valid signal on line 62 are used to control a pass gate 52, a pass gate 53, a pass gate 54 and a pass gate 55 within the metastable proof flip-flop. Pass gates 52 through 55 are placed in the metastable proof flip-flop to prevent the flip-flop from latching on a value of the input signal on D input 30 when the value of the input signal on D input 30 is not at a valid voltage (e.g., is not between 1.5 volts and 3.5 volts).

Operation of the metastable proof flip-flop shown in FIG. 10 is identical to the normal operation of the flip-flop. This does not hold for the case where the rising edge of clock signal (Clk) 41 occurs when the input signal on D input 30 is not at a valid voltage (e.g., is not between 1.5 volts and 3.5 volts). In this case, pass gates 52 through 55 will delay the capture of the data value on D input 30 until the input signal on D input 30 is at a valid voltage (e.g., the voltage on D input 30 is less than 1.5 volts or more than 3.5 volts). This prevents metastable states from occurring in the metastable proof flip-flop.

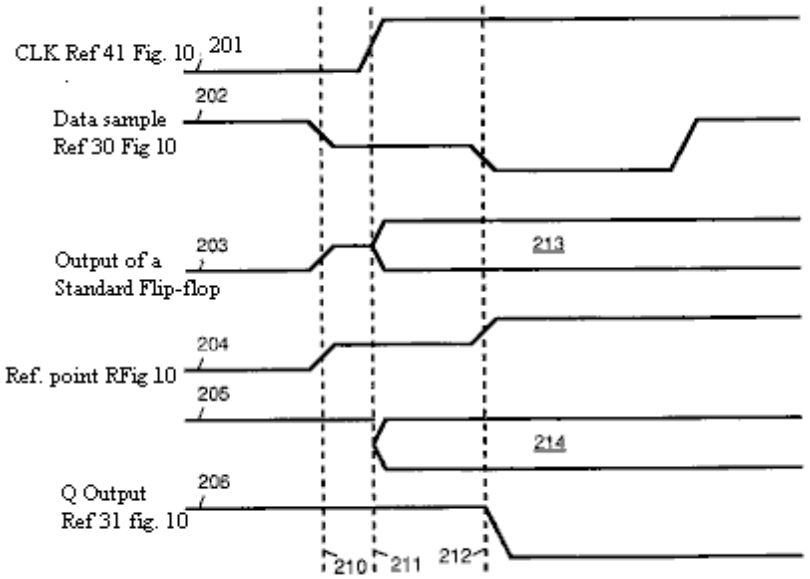

Figure 11: Timing diagram of metastable proof flip-flop

Figure 11 is a timing diagram which compares operation of a metastable proof flip-flop with a conventional flip-flop. A clock signal 201 represents a clock signal for a flip-flop, for example, clock signal (Clk) 41 of the metastable proof flip-flop shown in FIG.10. A data signal 202 represents a signal on a D input of a flip-flop. D input 30 of the metastable proof flip-flop shown in FIG.10. At a time 210, data signal 202 transitions to a voltage which is less than 3.5 volts and is greater than 1.5 volts. At a time 211, clock signal 201 transitions from a low voltage state (0 volts) to a high voltage state (5 volts). At a time 212, signal 202 transitions from the voltage which is less than 3.5 volts and is greater than 1.5 volts to a low voltage state (0 volts).

When clock signal 201 transitions from a low voltage state (0 volts) to a high voltage state (5 volts) at time 211, data signal 202 is at a voltage which is less than 3.5 volts and is greater than 1.5 volts.

The result is that the captured data value is unknown and possibly metastable, as represented by uncertain region 213.

When clock signal 201 transitions from a low voltage state (0 volts) to a high voltage state (5 volts) at time 211, data signal 202 is at a voltage which is less than 3.5 volts and is greater than 1.5 volts and the voltage captured at reference node 24 is unknown and possibly metastable, as discussed above. When this voltage is gated to Q output 11 of the flip-flop, at time 211, the Q output signal is also unknown and possibly metastable, as represented by uncertain region 214.

A reference node signal 204 shows a voltage of a reference node within a metastable proof flip-flop, for example at node 63 of the metastable proof flip-flop shown in FIG.10. When clock signal 201 transitions from a low voltage state (0 volts) to a high voltage state (5 volts) at time 211, pass gates 52 through 55 will delay the capture of the data value on D input 30 until the input signal on D input 30 is at a valid voltage (e.g., the voltage on D input 30 is less than 1.5 volts or more than 3.5 volts). This prevents metastable states in the metastable proof flip-flop. At time 212, when signal 202 transitions from the voltage which is less than 3.5 volts and is greater than 1.5 volts to a low voltage state (0 volts), the metastable proof flip-flop captures a valid voltage.

A Q output signal 206 shows a voltage at a Q output of a metastable proof flip-flop, for example, at Q output 31 of the metastable proof flip-flop shown in FIG.10. When clock signal 201 transitions from a low voltage state (0 volts) to a high voltage state (5 volts) at time 211, pass gates 52 through 55 will delay the capture of the data value on D input 30 until the input signal on D input 30 is at a valid voltage (e.g., the voltage on D input 30 is less than 1.5 volts or more than 3.5 volts). As discussed above, this prevents metastable states in the metastable proof flip-flop. At time 212, when signal 202 transitions from the voltage which is less than 3.5 volts and is greater than 1.5 volts to a low voltage state (0 volts), the metastable proof flip-flop captures a valid voltage. This captured voltage is passed through, at time 212 to Q output 31, as illustrated by Q output signal 206.

Shortly, a metastable proof flip-flop receives an input value on flip-flop input. The flip-flop holds an output value on a flip-flop output. In response to a transition of a clock signal, a transition in the output value occurs. The new output value is the input value formerly received by the flip-flop. In order to make the flip-flop metastable proof, the transition in the output value is delayed when the input value is in a metastable state. When the input value is no longer in the metastable state, then the transition in the output value is allowed to complete.

Using the notion of latched feedback memory, Ronald Freyman (2) has invented a novel concept a latch circuit with reduced metastability. A latch circuit employs a feedback arrangement comprising a transmission gate circuit that conducts only when the output node is in the mid-voltage state. At the onset of a metastable state, the feedback arrangement forces a receiving node into its previous stable state. This eliminates or reduces the possibility that the latch could remain hung for an indefinite period in a metastable state.

The use of the metastable proof flip-flop as the input guard latch in the Letcher state machine can make the Letcher state machine metastable proof.

Uses of Letcher State Machines in Commercial Products

The Letcher State Machine has been widely made use of in digital circuit designs and CPU design techniques by many hardware companies for their various commercial products. It has been used effectively in many simple and complex systems.

Cyrix Corporation has used the basic Letcher State Machine in implementing many of its designs for Ultra high-speed arithmetic units. Some of the prominent patents assigned to Cyrix Corporation were:

1) "Method and apparatus for performing division using a rectangular aspect ratio multiplier," by David Matula[10].
2) "Method and apparatus for performing the square root function using a rectangular aspect ratio multiplier," by Willard Briggs[11].

Both Matula and Briggs have made use of the Letcher Finite State Machine, which is metastable proof, in their respective Patents for implementing the mathematical operations. The entire algorithm for calculating the rectangular aspect ratio multiplier and the square root function acts as the function block for a single Letcher State Machine.

The arithmetic unit is one of the most important components of any integrated electronic data processing system. Arithmetic units perform a wide variety of mathematical functions upon operands, which are transmitted from other portions of an integrated system. The basic addition, subtraction, and multiplication functions are quickly and efficiently performed in arithmetic units today. However, presently available techniques for performing division functions have not been completely satisfactory with respect to efficiency and speed.

Briggs and Matula have overcome this deficiency by making use of the Letcher State Machine in their designs. Figure 12 shows a block diagram of the arithmetic system of their invention. According to this diagram, The system bus 32 includes the data line, address line and control line. These input signals from the system bus are provided to a D latch 34, a digit latch 36 and a reciprocal latch 33. The output signals of these latches are provided to the arithmetic logic unit. The logic unit has two output sets. An output set is provided to the feedback register 54 and the input to the logic unit. Another output set is provided to the E latch 38. The E latch 38 produces output signals. This system is a typical Letcher state machine as we have a function module surrounded by latches. It has all advantages of a Letcher state machine in terms of efficiency, speed, and no possibility of metastability failure.

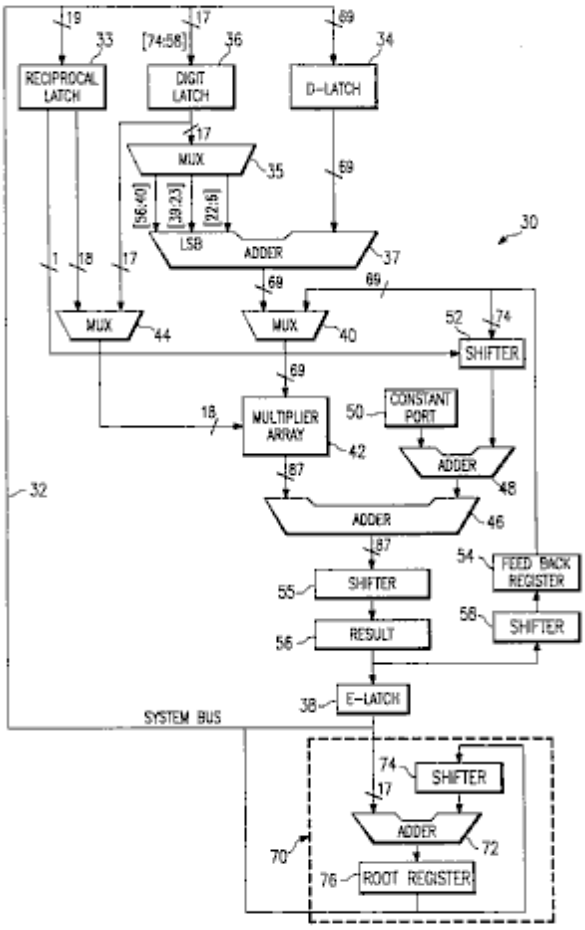

Figure 12: Arithmetic unit using Letcher State Machine

Cypress Semiconductor Corporation is also using the Letcher Finite State Machine in designing their PLD's (Programmable Logic Devices). The Cypress CY7C330[12] is a high performance, erasable, programmable, logic device (EPLD) whose architecture has been optimized to enable the user to easily and efficiently construct very high performance synchronous state machines that can communicate either with each other or with microprocessors over bi-directional parallel buses of user-definable widths.

The Logic Block Diagram of the CY7C330 (Fig 13) shows a number of latches surrounding a combinatorial logic where the state registers of arbitrary type feedback to a combinatorial array. This is typical of a Letcher State Machine. The CY7C330 features a pipelined architecture that achieves a 66-Mhz state transition speed. The device's 11 dedicated register inputs offer small set-up and hold times. The device has 12 I/O macrocells, which are used for inputting and outputting information from the programmable logic array.

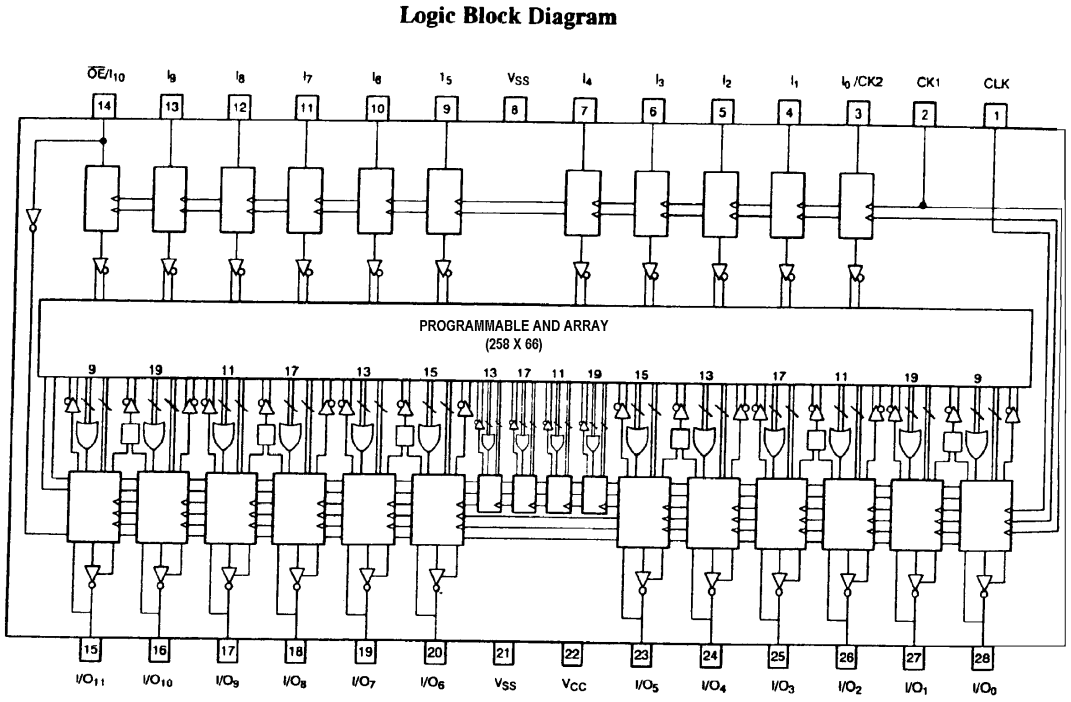

Figure 13: Logic Block Diagram of Cypress CY7C330

Jagdish Pathak[13] (Cypress Corporation) in his patents, improvised on the existing macrocells by providing a dual I/O macrocell which helped in increasing the speed of the synchronous state machines. He has also made use of the Letcher State Machine in his designs.

Ronald L. Freyman[14], in his patent assigned to AT&T Bell Laboratories has made use of the Letcher State Machine in designing a Latch Circuit with reduced metastability. The latch circuit employees a feedback arrangement with the transmission gate circuit. At the metastable state, the feedback arrangement forces it into the previous state and thus reduces metastability.

Ronald A. Olisar[15], in his patent assigned to Tektronix, Inc., has also made use of the Letcher State Machine wherein he talks about the improvement in Input and Output selector logic. He makes use of a multiplexer along with the logic elements to select the logic elements alternately at every positive and negative half cycles of the clock. This helped enhance the speed of the operation.

Results and Conclusion

The design of central processor units, cache controllers, and arithmetic units can be significantly simplified by the use of Letcher State Machines. Since the original patent and publication of the design of the state machine, changes in technology and new innovations have resulted in new applications of the state machine, and in processes and hardware that make implementing the state machine easier. Throughout the twelve years of its existence, the Letcher State Machine has been used in numerous applications, with no reports of metastability. Future changes in latch design and in other areas promise to make the Letcher State Machine even more useful, as more and more applications take advantage of the state machines unique advantages in the area of massively parallel operations.

The three latches of the Letcher machine need not be identical in construction. Only the Input Latch in Figure 5 must be metastable proof as the other two latches are presented with signals that fully comply with the timing constraints for stability. Clearly, the Feedback Latch must have the lowest propagation delay of the three to obtain the fastest clock rate, as the maximum clock rate is the reciprocal of the propagation delay of the Function module plus the propagation delay of the Feedback Latch. Allowing a slower Output latch could be useful in superpipelined processors when the output signals are not needed until a fraction of the clock cycle later.

Areas for continued research include basic logic cell designs and implementation, new and faster latch designs, and the areas of metastable proof latches mentioned above.
Smaller feature sizes and lower voltages have continued to decrease the maximum propagation times for the technologies used to make the combinatorial logic blocks, and research in this area will continue to increase the potential maximum speed of Letcher State Machines. Research in the area of faster, more stable latches will continue to increase the maximum speed of Letcher state machines.

All of these areas offer potential for continued improvements in the finite state machine, which will result in faster processors. As machines become simpler and faster, we can expect more and more complex processors, performing complex tasks faster than we would ever have dreamed possible.

[1] Comer, David J. Digital Logic and State Machine Design. Second Edition, Saunder College Publishing 1990.

[1]Kenneth J Breeding, "Digital Design Fundamentals," Prentice Hall, 1992

ibid.

[3] J.H. Letcher. "Latched Feedback Memory Finite State Engine." Patent Number 4786829. U.S. Patent and Trademark Office. (1998)

[4] Katz, Randy H. Contemporary Logic Design. Addison Wesley Publishing Company, 1993.

[5] Olisar, et al. "High speed state machine." Patent Number 4873456. U.S. Patent and Trademark Office. (1989).

[6] Strong; Richard M. "Double-edge triggered memory device and system." Patent Number 5250858. U.S. Patent and Trademark Office. (1993).

[7] Keech, Eugene. "Metastable Immune Flip-Flop Arrangement." Patent Number 4963772. U.S. Patent and Trademark Office. (1990)

[8] Wagener, Peter. Metastability, A Designer's Viewpoint. Boeing.

[9] Baumann, Douglass, et. al. "Metastableproof flip-flop." Patent Number 5754070. U.S. Patent and Trademark Office. (1998)

[10] Matula, David, et al. "Method and apparatus for performing division using a rectangular aspect ratio multiplier." Patent Number 5046038. U.S. Patent and Trademark Office. (1991).

[11] Briggs, et al. "Method and apparatus for performing the square root function using a rectangular aspect ratio multiplier." Patent Number 5060182. U.S. Patent and Trademark Office. (1991).

[12] Cypress Semiconductor Data Book. Cypress Semiconductor, 1992.

[13] Pathak, Jagdish. "Architecture of High Speed Synchronous State Machine." Patent Number 5023484. U.S. Patent and Trademark Office. (1991)

[14] Freeman, Ronald. "Latch Circuit with Reduced Metastability." Patent Number 4963772. U.S. Patent and Trademark Office. (1992)

[15] Olisar, 1989.