N94-18378
12.3.1

Improved Self Arbitrated VLSI Asynchronous Circuits

P. Winterrowd
NASA Space Engineering Research
Center for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract- **This paper introduces an improved method for designing the class of CMOS VLSI asynchronous sequential circuits introduced in the paper by Sterling R. Whitaker and Gary K. Maki, "Self Arbitrated VLSI Asynchronous Circuits."**

1 Introduction

Synchronous sequential circuits are often the first choice in VLSI design. Races are avoided by synchronizing the circuit with a common clock signal; however, the frequency of this signal must be slow enough to allow signals to propagate through the slowest block regardless of how often that block's output is actually used. Also, the RC delays introduced in VLSI design make synchronizing the circuit with a clock signal increasingly difficult as the circuit's complexity increases. Moreover, with CMOS circuits peak power usage is attained during switching. If several blocks are synchronized by a clock signal and, thus, switching at the same time then the peak power required by the chip is greatly increased.

These limitations can be avoided by designing with asynchronous circuits. The paper by S. Whitaker, "Self Arbitrated VLSI Asynchronous Circuits", presents an asynchronous circuit with some interesting qualities. Of main interest here is the simple design by inspection rules that arise from this circuit. This paper presents a variation on Whitaker's circuit which reduces the number of transistors required.

2 Circuit Model

The general model for this circuit is the same as that given in Whitaker's research. There is an enable and disable block feeding into a buffer stage as shown in Figure 1.

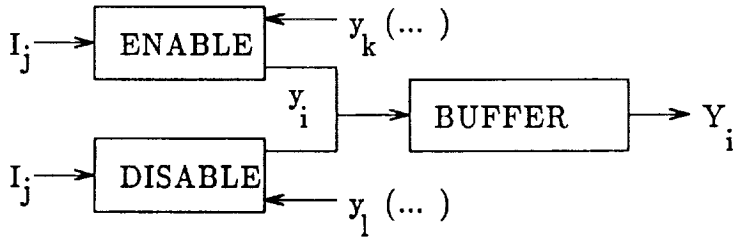

Figure 1: General Circuit Model

Where y_i and Y_i are present and next state variables respectively, and I_j represents input signals.

The variation on the original circuit presented here differs in how the buffer, enable, and disable blocks are implemented. Whitaker's buffer circuit consisted of two inverters and two weak feedback transistors as shown in Figure 2:

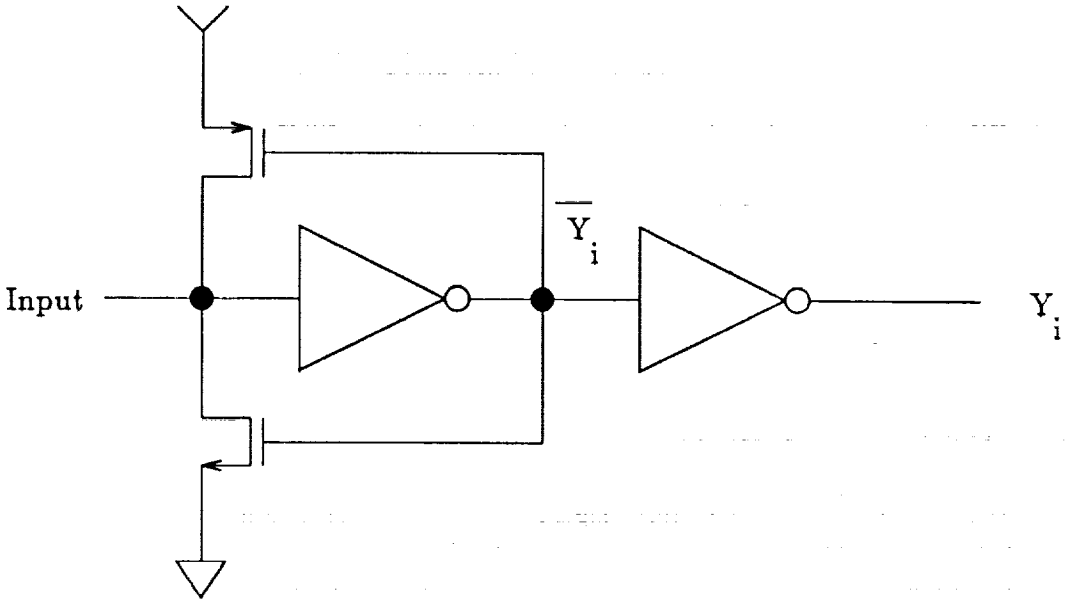

Figure 2: Original Buffer Circuit

As shown, this buffer circuit provides not only Y_i but also $\overline{Y_i}$. The buffer state table for this circuit is given in table one.

Table 1: Original Buffer State Table

y_i	Input	Y_i
0	0	0
0	1	1
1	0	0
1	1	1
0	Z	0
1	Z	1

The buffer for the circuit presented in this paper is shown in Figure 3. Although it only produces the Y_i variable and not it's complement, it saves two transistors and in the design procedure for this circuit the complemented variable is not needed. The buffer state table for this circuit is given in Table 2.

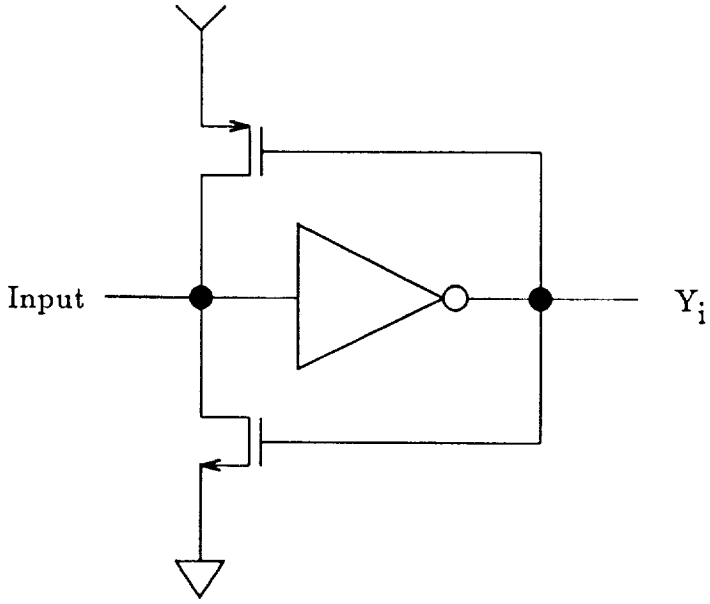

Figure 3: New Buffer Circuit

The enable and disable blocks are simple pass networks and their design is completely specified by the design equations given later in this paper.

Table 2: New Buffer State Table

y_i	Input	Y_i
0	0	1
0	1	0
1	0	1
1	1	0
0	Z	0
1	Z	1

3 State Assignment

The state assignment for a flow table remains the same in this paper as in its predecessor, a simple one-hot-code state assignment as shown in Table 3.

Table 3: Example Flow Table

	I_1	I_2	I_3	y a	y b	y c	y d	y e	y f
A	(A)	E	F	1	0	0	0	0	0
B	A	(B)	F	0	1	0	0	0	0
C	A	B	(C)	0	0	1	0	0	0
D	(D)	B	C	0	0	0	1	0	0
E	D	(E)	C	0	0	0	0	1	0
F	D	E	(F)	0	0	0	0	0	1

These circuits effect a non-normal transition. Thus, in order to show that the circuit operates correctly, it is necessary to show that no transition path between two states overlaps any transition path between two other states.

Definition 1: *Let $[y_i]$ be the state with bit y_i set and let $[y_a\ y_b\ y_c...]$ be the binary number with the appropriate number of bits with the y_a, y_b, y_c... bits set.*

Definition 2: *By definition (to be shown later) let a proper transition path between $[y_i]$ and $[y_j]$ be $[y_i\ y_j]$. Note that from the one hot code state assignment every state can be expressed by $[y_k]$, where y_k represents the one bit which should be set for any state.*

Theorem 1: *Given a one-hot-code state assignment with the above "proper" transition path, no two transition paths will overlap.*

Proof: Since the transition path for two states $[y_i]$ and $[y_j]$ is given by $[y_i\ y_j]$ this will overlap with the transition $[y_a]$ and $[y_b]$ given by $[y_a\ y_b]$ only if $y_a = y_i$ and $y_b = y_j$; thus, the transition between two states only overlaps itself.

Therefore, to show that the circuit presented here correctly implements a flow table it must simply be shown that it correctly implements the "proper" transition path referred to above.

4 Design Procedure

This section starts with a definition.

Definition 3: *Let a scale-of-two of loop in a flow table be defined by a any state A under input I_k which goes to state B under input I_j where state B returns to state A under input I_k.*

The basic design equations for a circuit can be expressed as:

$$\Sigma(I_j \Sigma y_k(0)) + \Sigma y_l(1) \tag{1}$$

Enable Expr. + Disable Expr.

Note that the disable expression form only holds in the absence of order-of-two loops.

Looking at an example flow table as given in Table 3 and reproduced here for convenience:

Table 3: Example Flow Table

	I_1	I_2	I_3	y y y y y y a b c d e f
A	(A)	E	F	1 0 0 0 0 0
B	A	(B)	F	0 1 0 0 0 0
C	A	B	(C)	0 0 1 0 0 0
D	(D)	B	C	0 0 0 1 0 0
E	D	(E)	C	0 0 0 0 1 0
F	D	E	(F)	0 0 0 0 0 1

For each state the design procedure is simple:

For state $[y_i]$:

1. Identify all input states I_j under which $[y_i]$ is stable. These input states become the I_j's in the basic equation.

2. For each I_j identify all unstable states $[y_i]$ in the same column and note the stable state's row $[y_k]$ under which they occur. Again, these $[y_k]$'s become the y_k's in the basic equation under the appropriate I_j's.

3. Thus, the enable equation can be written as:

$$\Sigma(I_j \Sigma y_k(0)) \tag{2}$$

4. Identify all unstable states $[y_l]$ under the row for the stable state $[y_i]$.

5. The disable expressions for the state $[y_i]$ can be written as:

$$\Sigma y_l(1) \tag{3}$$

6. For each term in the disable expression determine if it is a member of a scale-of-two loop. If so, include the input state under which this term occurs as part of the term. For example, if $y_l(1)$ was under I_k and was a member of a scale-of-two loop, then $y_l(1)$ would become $I_k y_l(1)$.

As an example, the enable expression for state A, $[y_a]$, would be:

$$I_1(y_b(0) + y_c(0)) \tag{4}$$

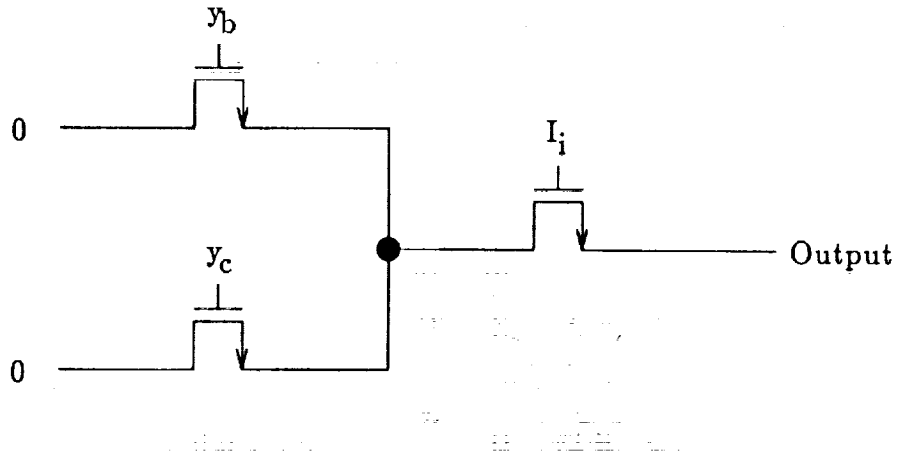

Figure 4: Example Enable Block

And, the disable expression for state A would be:

$$y_e(1) + y_f(1) \tag{5}$$

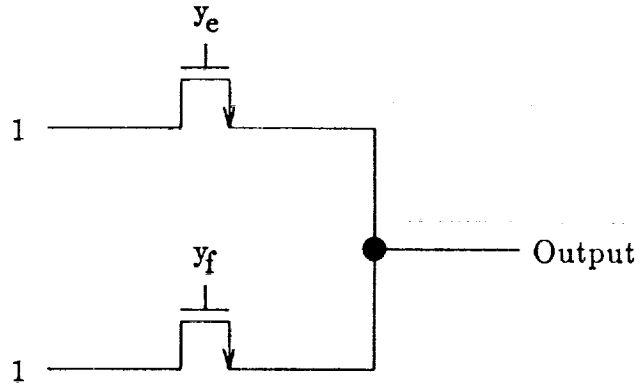

Figure 5: Example Disable Block

5 Circuit Operation

In order to show that this circuit operates correctly, it must be shown that the circuit transitions properly. To do this it must be shown that in going from state $[y_i]$ to $[y_j]$ under input I_j the transition path is $[y_i\ y_j]$.

Theorem 2: *Only positive and not complemented variables are used in the design equations for this circuit.*

Proof: This follows directly from the design procedure and the basic design equation.

Theorem 3: *The enable and disable equations for $[y_i]$ do not contain the present state term y_i.*

Proof: The design equations for state $[y_i]$ derive their terms from the unstable states in row $[y_i]$ and the stable states in the other rows which contain unstable $[y_i]$ states. Since every state $[y_i]$ in the row for the state $[y_i]$ is stable and every state $[y_i]$ not in the row for state $[y_i]$ is unstable, then the y_i term never appears in the design equations for state $[y_i]$.

Theorem 4: *For a reduced row flow table, for each unstable state $[y_u]$ under input I_u on the stable state $[y_s]$'s row that unstable state will contribute only the following terms to the design equations for that table:*

$$y_u(1) \text{ or } I_u y_u(1) \text{ disable term for } Y_s$$
$$I_u y_s(0) \text{ enable term for } Y_u$$

Proof: The proof follows directly from the design procedure.

Theorem 5: *For a reduced row flow table that transitions from state $[y_i]$ to the state $[y_k]$ under input I_j then while the circuit is in state $[y_i]$ under input I_j the only variable to be affected is Y_k which is set high.*

Proof: If this theorem were not true then either 1) Y_i would be set to 0 from this state, or 2) Y_k would not be changed, or 3) Some other variable (all of which are zero at this point due to one-hot-code state assignment) would be raised to a 1.

First, for Y_i to be set to zero with all other state variables at zero then the design equation would have to contain the term $y_i(1)$, which is invalid from Theorem 3, or $\overline{y_k}(1)$, which is invalid from Theorem 2.

Second, Y_k will be set high from Theorem 4 and the design procedure which state that the enable expression for Y_k will include $I_j y_k(0)$. Note that a conflict could occur if the design equation for Y_k contained $y_i(1)$; however, this would be a scale-of-two loop and the $y_i(1)$ term would be $I_k y_i(1)$ where I_k is different from I_j. Thus, there would be no conflict for a flow table properly designed and Y_k will be set high.

Finally, for another variable $[y_m]$ to be set high from the state $[y_i]$ under I_j then it's enable equation would have to contain the term $I_j y_i(0)$ which would, from the design procedure, mean that under the row for state $[y_i]$ under I_j would be state $[y_m]$; however, by definition this location contains $[y_k]$.

Thus, the theorem must be true since all other alternatives are false.

Theorem 6: *If the circuit is in the state $[y_i\ y_k]$ under input I_j where state $[y_i]$ goes to state $[y_k]$ under input I_j then the only variable to be affected is Y_i which is set low.*

Proof: If this theorem were not true then either 1) Y_i would stay high, or 2) Y_k would be set low, or 3) Some other variable would be set high (since all the other variables are, by definition, low to begin with.)

First, from Theorem 4 and the design procedure the design equations for Y_i contains the term $y_k(1)$ or $I_j y_k(1)$; thus, it would remain only not go low if it also included $I_j y_i(0)$, which is invalidated by Theorem 3, or $I_j \overline{y_l}(0)$, which is invalidated by Theorem 2, or $I_j y_k(0)$ which can be invalidated by the following argument. If the enable expression for

Y_i contained any I_j terms then the position under I_j would have to contain $[y_i]$ and, by definition, it contains $[y_k]$. Thus, Y_i is set low.

Second, for Y_k to be set low it would contain the term $y_k(1)$ or $I_j y_k(1)$, which can be invalidated from theorem three, or $\overline{y_l}(1)$, which can be invalidated by theorem two, or $y_i(1)$ which can be invalidated by the following argument. If Y_k did contain $y_i(1)$ then that would be a scale-of-two loop and the $y_i(1)$ term would have to be $I_m y_i(1)$ where I_m is not equal to I_j; thus, Y_k will not be set low.

Finally, for another variable $[y_m]$ to be set high from the state $[y_i\ y_j]$ under I_j, the enable equation would have to contain either the term $I_j y_i(0)$ or $I_j y_j(0)$. The term $I_j y_i(0)$ would, from the design procedure, mean that under the row for state $[y_i]$ under I_j would be state $[y_m]$ which by definition contains $[y_k]$. The term $I_j y_j(0)$ would, from the design procedure, mean that under the row for state $[y_j]$ under I_j would be state $[y_m]$ which by definition contains the stable state $[y_j]$. Thus, no other variable will be set high.

Therefore, since all the other alternatives are proven false, this theorem must be true.

So from Theorems 5 and 6 the circuit created from the design procedure in the previous section fulfills Theorem 1 and is critical-race free.

6 Transistor Count Comparison

From the assignment procedure it is obvious that we need one state variable from each state. Also, from Figure 1, which shows the general circuit model for this circuit, it is clear that for each state variable we need four transistors (one buffer.) Moreover, from Theorem 4 and the design procedure it can be shown that for each stable state $[y_i]$ in a flow table one transistor is required if an unstable state $[y_i]$ is also in that column and for every unstable state two transistors are required. Also, every scale-of-two loop contributes an additional two transistors. Thus, the number of transistors needed for a reduced row flow table is given by:

$$Total\ \#\ of\ Transistors = 4S + B + 2U + 2L \tag{6}$$

Where s = number of states, b = number of stable states in the low table with identical unstable states in the same column, u = number of unstable states in flow table, and L = number of scale-of-two loops that exist in the flow table. The number of transistors for the design method in Whitaker's paper can be shown to be:

$$Total\ \#\ of\ Transistors = 6S + 4U + 2L \tag{7}$$

Thus, for the example flow table given in Table 3 where s=6, b=6, u=12, and L=0 the total number of transistors for the improved design method is 54, and the total number for the old method is 84; thus, a difference of 30 transistors.

Table 4 shows some typical values for reduced row flow tables and compares transistor counts.

Table 4: Transistor Count Comparison

S	B	U	L	Transistor Count		New/Old
				Old Method	New Method	
6	6	12	0	84	54	0.643
8	9	15	1	110	73	0.664
7	10	18	2	118	78	0.661
9	10	17	0	122	80	0.656

Reset Feature

In any circuit, asynchronous or synchronous, it is advantageous to be able to preset the circuit to some state. This is almost a necessity upon startup since a circuit often begins in an unknown or undesirable state. The basic model for this circuit can be modified to easily include a reset feature as shown in figure six:

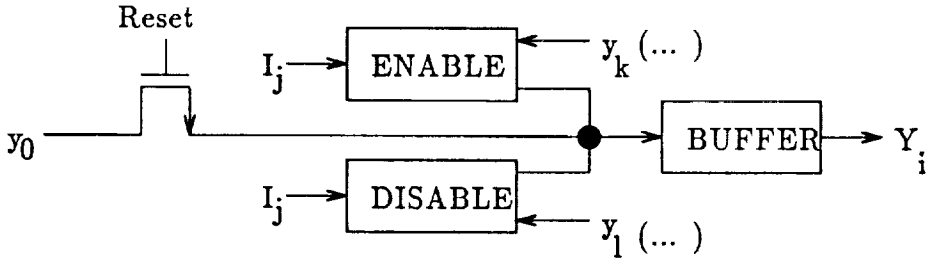

Figure 6: Modified General Circuit Model

This feature only costs one transistor for each state; however, the design must insure that all inputs are low while the reset is high.

Summary

The general circuit model for this paper and the one-hot-code state assignment lead to an easily designed and implemented asynchronous circuit. Once an input is introduced the circuit sets the new variable high which then sets the variable signifying the old state low. This $[y_i];[y_i\ y_j];[y_j]$ non-normal mode transition insures that no two transition paths overlap; thus, the circuit is critical-race free.

This improved design method reduces the transistor count from the old method by, roughly, one third, decreasing the size of the overall circuit and increasing its usefulness.

Finally, although the circuit operates at 1/2 the speed of an STT state assignment asynchronous circuit due to its non-normal mode operation it is very easy to design and avoids the disadvantages of a synchronous circuit such as clock routing, power bussing, and speed dependency upon slowest information path.

References

[1] S. Whitaker, S. Manjunath and G. Maki, Self Arbitrated VLSI Asynchronous Circuits, *NASA SERC 1990 Symposium on VLSI Design*, pp. 87-103.

[2] S. Golpalakrishnan, G. Kim and G. Maki, Implications of Tracey's Theorem to Asynchronous Sequential Circuit Design, *NASA Symposium on VLSI Design*, pp. 9.1.1-9.1.11, Nov. 1990.

[3] S. Whitaker and G. Maki, Pass Transistor Asynchronous Sequential Circuits, *IEEE Journal of Solid State Circuits*, pp. 71-78, Feb 1989.

This research was supported (or partially supported) by NASA under Space Engineering Research Center Grant NAGW-1406.