Synthesis of Saturating Counters Using Traditional and Non-traditional Basic Counters

Zhaojun Wo and Israel Koren
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

Abstract

Saturating counters are a newly defined class of generalized parallel counters that provide the exact number of inputs which are equal to 1 only if this number is below a given threshold. Such counters are useful in, for example, self-test and repair units for embedded memories. This paper defines saturating counters for arbitrary threshold values and presents several alternatives for their implementation. The delay and area of the proposed design alternatives are then estimated using a 0.25μm cell library. Finally, we study the behavior of saturating counters when the threshold approaches the number of input bits, i.e., the special case of non-saturating parallel counters.

1. Introduction

Parallel counters are used in many designs (such as multipliers) and consequently have had many different implementations (e.g., [1, 2]). These implementations use various basic building blocks such as (3,2) counters, (7,3) counters and the like [3]. An (n,k) parallel counter is defined as having n input bits and producing a k-bit binary count of its "1" inputs, with $2^k - 1 \geq n$ or $k \geq \lceil \log_2(n+1) \rceil$. In [4], we defined a new type of parallel counter which we called a *saturating counter*. A saturating counter provides the exact count of its inputs that are 1 only if this count is below a certain threshold, denoted by T. When the number of inputs that are 1 is either equal to or exceeds the threshold T, the output indicates only that the threshold has been reached and does not supply the exact count. This type of counter is useful in the design of a self-test and repair circuit for large memories (about 1024 bits wide) embedded in a system-on-a-chip [4]. Such a circuit counts the number of faulty bits in a row and if this number exceeds the number of spare memory bits, denoted by g, the memory is not repairable. Therefore, it is sufficient to know the failing bit count accurately only up to g - the number of spares available, and we need to design a fast saturating counter with a threshold $T = g + 1$.

Note that the saturating counters considered here are different from those used in certain image processing applications and in microprocessors' branch prediction units; the latter normally saturate at their maximum count of $2^k - 1$ (and sometimes also at their minimum count of 0), and all other results must be exact. Also note that when the threshold T is equal to the number of input bits n, the saturating counter turns into a regular parallel counter.

The number k of output bits of a saturating counter with n inputs and a threshold T will be, in most cases, considerably smaller than the number required for the corresponding parallel counter. Instead of $k = \lceil \log_2(n+1) \rceil$ we set $k = \lceil \log_2 T \rceil + 1$. In [4] we limited the threshold T to powers of 2. In this paper we extend the threshold values to include any number smaller than n. For example, if $n = 1024$, a parallel counter would require 11 output bits while a saturating counter with a threshold of $T = 12$ uses only 5 output bits.

Note that we set k to be $\lceil \log_2 T \rceil + 1$ rather than $\lceil \log_2(T+1) \rceil$, which would be the number of output bits of a parallel counter capable of counting up to T input bits. If T is a power of 2 then $k = \lceil \log_2 T \rceil + 1 = \lceil \log_2(T+1) \rceil$; otherwise $k = \lceil \log_2(T+1) \rceil + 1$. For example, for $T = 4$ we set $k = 3$ but for $T = 5$ we set $k = 4$. As will become evident in Section 2, very simple implementations of saturating counters are obtained when the most significant output bit of the counter is made into a "sticky bit", i.e., this bit is the result of an OR operation of all carries into this bit position. Thus, for $T = 4$ we can set $k = 3$, implying that the most significant output bit will be of weight 2^2, providing an accurate output if the number of "1" inputs is no more than 3. However, if we would use for $T = 5$ only $k = 3$, the fact that the most significant output position of weight 2^2 behaves as a sticky bit may result in a situation where for eight "1" inputs the output will be 4, which is below the threshold and is thus considered a valid count. Therefore, $k = 4$ must be used in order to avoid such situations.

2. Saturating Counters - Design Alternatives

An $[n, T]$ saturating counter has n input bits denoted by $a_1, a_2, \cdots, a_n$ and a threshold of $T < n$. The counter has k output bits denoted by $s_{k-1}, s_{k-2}, \cdots, s_0$ where $k = \lceil \log_2 T \rceil + 1$. The output satisfies

$$(s_{k-1}, s_{k-2}, \cdots, s_0)_2 = \sum_{i=1}^{n} a_i$$

$$\text{if} \quad \sum_{i=1}^{n} a_i \leq T - 1$$

while

$$(s_{k-1}, s_{k-2}, \cdots s_0)_2 \in [T, 2^k - 1]$$

$$\text{if} \quad \sum_{i=1}^{n} a_i \geq T$$

For example, a $[1024, 7]$ saturating counter has 1024 inputs, a threshold of 7, and produces four output bits satisfying

$$(s_3, s_2, s_1, s_0)_2 = \sum_{i=1}^{1024} a_i$$

if at most six input bits are equal to 1, and $(s_3, s_2, s_1, s_0)_2 \in [7, 15]$ if seven or more input bits are 1.

A complete Wallace tree for 1024 inputs produces eleven output bits and requires 16 levels of (3,2) counters. A straightforward way to implement a $[1024, 7]$ saturating counter is to use (3,2) counters in the columns with weights $2^2, 2^1$ and 2^0 but use only OR gates in the column with weight 2^3. This implementation makes the most significant output bit a sticky bit, which, once set to 1, will never change back to 0 during the remaining steps of the operation. The above implementation, shown in Table 1, requires 11 levels of (3,2) counters plus one level of an OR gate, assuming that two levels of OR operations in column 2^3 can be completed in parallel to the operation of a single level of (3,2) counters in the $2^2, 2^1$ and 2^0 columns. Table 1 shows, for each level of the tree, the number of (3,2) or (2,2) counters required in every column, and the resulting number of intermediate results in every column. For example, in the second level of the tree, 114 (3,2) counters are used in the 2^0 column, producing 114 intermediate bits of weight 2^0 and 114 bits of weight 2^1, which are added to the 115 bits generated directly in the 2^1 column. The notation 9+2(OR_4) in the 2^3 column means that two levels of OR gates are used, 9 in the first level and 2 in the second.

Note that the implementation depicted in Table 1 will produce a result of 8 if the number X of "1" input bits satisfies $X mod\, 8 = 0$, e.g., 16, 32 and so on. This, however, is not a problem since the saturating counter is expected to produce the correct count only if the number of 1's at the inputs is less than or equal to 7. Also note that if $T < 2^{k-1}$ ($T < 8$ in the example shown in Table 1), we may set $T = g$ instead of $T = g + 1$, where g is the number of available spare memory bits, since a counter that provides an exact count for all values up to T will do so also for $T + 1$ as long as $T + 1 \leq 2^{k-1}$. Only when g is a power of 2 must we set $T = g + 1$. Table 1 was generated using an online saturating counter simulator which is available at [6].

In [1], Jones and Swartzlander compare the design of parallel counters using only (3,2) or (2,2) counters to designs using more complex counters like (7,3), (15,4) and (31,5). They analyze the delay and area of different implementations and conclude that designs based on only (3,2) and (2,2) counters are generally superior. We therefore decided not to experiment with counters like (7,3), (15,4) and (31,5). However, in recent years, (4;2) compressors [3] have become common in parallel multiplier designs, and very efficient implementations have been proposed for them (e.g., [5]). We studied in [4] the possibility of using (4;2) compressors instead of (3,2) counters in one or more levels of the saturating counter, concluding that their use will reduce the total number of levels. However, the delay of each level increases and thus, more accurate delay comparisons are needed. Those will be reported in Section 3.

We have also experimented with several non-traditional basic counters like (4,3), (5,3), and (8,4) which typically are not used in conventional parallel counters. Traditional basic counters are normally of the type $(2^k - 1, k)$ where all combinations of the k output bits are utilized. In contrast, a (4,3) counter for example, will never generate the outputs $101_2 = 5_{10}$, $110_2 = 6_{10}$ and $111_2 = 7_{10}$. Using (4,3) counters in the design of parallel counters does not make much sense. However, it may be useful for a saturating counter since 4 is the largest number of inputs which would still require only two levels of XOR gates as the (3,2) counter does. Table 2 shows that if (4,3) counters are used in levels 1 through 4, the total number of levels is reduced from 12 to 11 and more levels use only (2,2) counters which have a lower delay than (3,2) counters. The detailed comparison of these design alternatives is presented in Section 3.

The (5,3) counter with which we experimented, can be viewed as a modified (4;2) compressor with all of its 5 inputs arriving at the same time rather than having one input arrive later as a carry from a previous stage. More importantly, instead of generating two outputs of weight 2^{i+1} (where 2^i is the weight of the input bits), it generates one output of weight 2^{i+1} and another of weight 2^{i+2}. Such a basic counter will propagate its carries to higher weight columns (compared to a (4;2) compressor), reaching faster the sticky bit position which absorbs the carries. On the other hand, this (5,3) counter will most likely require a more complex design (compared to a (4;2) compressor) with a higher area overhead. An even more aggressive basic counter that may be considered for implementing satu-

2^3	2^2	2^1	2^0	Level
			341(3,2)	
		341	342	1
		113(3,2)	114(3,2)	
	113	114+115	114	2
	113	229	114	
	37(3,2)	76(3,2)	38(3,2)	
37	76+39	38+76	38	3
37	115	115	38	
9+2(OR$_4$)	38(3,2)	38(3,2)	12(3,2)	
4+38	38+39	12+39	14	4
42	77	51	14	
10+3(OR$_4$)	25(3,2)	17(3,2)	4(3,2)	
3+25	17+27	4+17	6	5
28	44	21	6	
7+1(OR$_4$)	14(3,2)	7(3,2)	2(3,2)	
4+14	7+16	2+7	2	6
18	23	9	2	
4+1(OR$_4$)	7(3,2)	3(3,2)	1(2,2)	
3+7	3+9	1+3	1	7
10	12	4	1	
2+1(OR$_4$)	4(3,2)	1(3,2)		
1+4	1+4	2	1	8
5	5	2	1	
1+0(OR$_4$)	1(3,2)	1(2,2)		
2+1	3+1	1	1	9
3	4	1	1	
0(OR$_4$)	2(2,2)			
3+2	2	1	1	10
5	2	1	1	
1(OR$_4$)	1(2,2)			
2+1	1	1	1	11
3	1	1	1	
1(OR$_4$)				12
1	1	1	1	

Table 1. A $[1024, 7]$ saturating counter using (3,2) and (2,2) counters. It uses 892 (3,2) counters, 5 (2,2) counters and 49 4-input OR gates.

2^3	2^2	2^1	2^0	Level
			256(4,3)	
	256	256	256	1
	64(4,3)	64(4,3)	64(4,3)	
64+64+64	64+64+64	64+64	64	2
192	192	128	64	
48+12(OR$_4$)	48(4,3)	32(4,3)	16(4,3)	
12+96+32	48+32+16	32+16	16	3
140	96	48	16	
35+8(OR$_4$)	24(4,3)	12(4,3)	4(4,3)	
11+48+12	24+12+4	12+4	4	4
71	40	16	4	
17+5(OR$_4$)	20(2,2)	8(2,2)	2(2,2)	
5+20	20+8	8+2	8+2	5
25	28	10	2	
6+1(OR$_4$)	14(2,2)	5(2,2)	1(2,2)	
4+14	14+5	5+1	1	6
18	19	6	1	
4+1(OR$_4$)	6(3,2)	2(3,2)		
3+6	7+2	2	1	7
9	9	2	1	
2(OR$_4$)	4(2,2)	1(2,2)		
3+4	5+1	1	1	8
7	6	1	1	
1+1(OR$_4$)	2(3,2)			9
1+2	2	1	1	
3	2	1	1	
	1(2,2)			10
4	1	1	1	
1(OR$_4$)				11
1	1	1	1	

Table 2. A $[1024, 7]$ saturating counter using (4,3), (3,2) and (2,2) counters. It uses 584 (4,3) counters, 10 (3,2) counters, 56 (2,2) counters and 142 4-input OR gates.

rating counters is the (8,4) counter which, with its 8 inputs, will still need only three levels of XORs to generate the least significant output bit. It will have a reduction rate of 2 (number of inputs to the number of outputs) and furthermore, will reach the sticky bit position fast.

Our experiments with the above two non-traditional basic counters have shown that although we can achieve further reduction in the overall delay of the saturating counter, the corresponding area requirements are too high to be acceptable. For the sake of brevity we do not report the results of our experiments with these two counters.

2.1. ($\{m, i, j\}, 3$) units

Re-examining Tables 1 and 2, one notices that the last few stages achieve only a small reduction in the number of bits but incur a high delay. The last four stages in Table 1, which reduce the number of bits from (5,5,2,1) to (1,1,1,1), can be replaced by a look-up table with 2^{5+5+2} inputs and 4 outputs. An even simpler and probably faster (for most technologies) solution exists which takes advantage of the saturating nature of the counter. This solution uses a special 3-column ($\{5,5,2\}$,3) unit, shown in Table 3, that has been proposed in [4]. Here we investigate a more aggressive design which would replace the last five stages in Table 1 by a ($\{10,12,4\}$,3) unit as shown in Table 4. We can apply the same approach to the [1024,7] saturating counter, which uses (4,3) counters (see Table 2), and replace the last four stages by a ($\{9,9,2\}$,3) unit.

An ($\{m, i, j\}, 3$) unit, shown in Figure 1, is a saturating parallel counter which receives m inputs of weight 2^{k-1}, i inputs of weight 2^{k-2} and j inputs of weight 2^{k-3}. It produces three outputs of weights 2^{k-1}, 2^{k-2} and 2^{k-3} where k is the number of output bits of the entire saturating counter.

The definition and implementations of such units in [4] were restricted to the case where $2 \leq j \leq 3$, for which the maximum carry from the position of weight 2^{k-3} to the position of weight 2^{k-2} is 1. We now allow j to be any value in the

2^3	2^2	2^1	2^0	Level
10	12	4	1	7
2+1(OR$_4$)	4(3,2)	1(3,2)		
1+4 5	1+4 5	2 2	1 1	8
1(OR$_4$)	({5,5,2},3)			9
1	1	1	1	

Table 3. The bottom stages of a $[1024, 7]$ saturating counter using a $(\{5, 5, 2\}, 3)$ unit (891 (3,2) counters, a (2,2) counter, 43 4-input OR gates and a $(\{5, 5, 2\}, 3)$ unit).

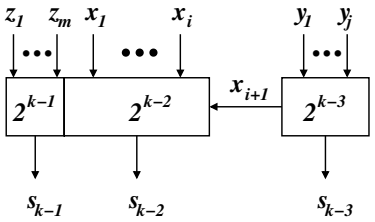

Figure 1. An $(\{m, i, j\}, 3)$ unit.

range $[2, 7]$, and as a result the maximum carry from the position of weight 2^{k-3} may be 3, i.e., a carry to the position of weight 2^{k-2} and another carry to the position of weight 2^{k-1}. Thus, we have $i + 1$ bits of weight 2^{k-2} and $m + 1$ bits of weight 2^{k-1} to be added. The Boolean expression for S_{k-3} is $s_{k-3} = y_1 \oplus \cdots \oplus y_j$. To derive the expressions for s_{k-2} and s_{k-1} we make the following observations.
The contribution (i.e., carry) of $y_1, \cdots, y_j$ to s_{k-2} can be represented by an additional input of weight 2^{k-2}, e.g.,

$$x_{i+1} = \begin{cases} y_1y_2 \text{ for } j = 2 \\ y_1y_2 + y_1y_3 + y_2y_3 \text{ for } j = 3 \\ y_1y_2 + y_1y_3 + y_1y_4 + y_2y_3 + y_2y_4 \\ \quad + y_3y_4 \text{ for } j = 4 \end{cases}$$

In general, there are $\begin{pmatrix} j \\ 2 \end{pmatrix}$ terms of the form y_uy_v.
The contribution (i.e., carry) of $y_1, \cdots, y_j$ to s_{k-1} can be represented by an additional input of weight 2^{k-1}:

$$z_{m+1} = y_1y_2y_3y_4 \text{ for } j = 4$$

In general, there are $\begin{pmatrix} j \\ 4 \end{pmatrix}$ terms of the form $y_uy_vy_wy_s$.
The truth table for the $(\{m, i, j\}, 3)$ unit is:

$z_1+\cdots+z_m+z_{m+1}$	$x_1 \cdots x_i\, x_{i+1}$	s_{k-1}	s_{k-2}
1	ϕ	1	ϕ
0	All 0's	0	0
0	A single 1	0	1
0	Two or more 1's	1	ϕ

2^3	2^2	2^1	2^0	Level
10	12	4	1	7
2+1(OR$_4$)	({10,12,4},3)			8
1	1	1	1	

Table 4. The bottom stage of a $[1024, 7]$ saturating counter using a $(\{10, 12, 4\}, 3)$ unit (886 (3,2) counters, a (2,2) counter, 39 4-input OR gates and a $(\{10, 12, 4\}, 3)$ unit).

The resulting Boolean expressions are:

$s_{k-2} = x_1 + \cdots + x_i + x_{i+1}$ and

$s_{k-1} = z_1 + \cdots + z_m + z_{m+1} + (x_1x_2 + x_1x_3 + \cdots + x_ix_{i+1})$

We can substitute x_{i+1} into the expressions for s_{k-2} and s_{k-1}. This would result in product terms with up to three literals in s_{k-1}, i.e., fan-in ≤ 3. Notice that the simplified Boolean equation for s_{k-2} may in fact produce s_{k-2}=1 even if the correct value is 0, but only if s_{k-1}=1. Therefore, any error will be in the exact value of the result above the threshold, which will have no impact of the proper operation of the counter.

3. Numerical Results

We have performed detailed synthesis of several saturating counters using the TSMC 0.25μ ASIC library. We first synthesized the basic cells like the (2,2), (3,2) and (4,3) counters, the (4;2) compressor and the OR$_4$ gate. The implementations of the first three are shown in Figure 2. These counters and the (4;2) compressor use XOR gates which are implemented using MUXs ([3, 5]) as shown in Figure 2. The resulting area and delay of the basic cells are shown in Table 5 where the last two columns indicate the delay and area relative to those of a XOR gate. In all the remaining figures the shown area and delay are relative to those of a XOR gate.
For all the basic cells except OR$_4$ the design shown in Table 5 has the smallest possible area and as a result, the largest delay. For the OR$_4$ gate we have selected a design that will satisfy the condition $\Delta_{OR_4} \leq 0.5\Delta_{(3,2)}$ to allow two levels of OR$_4$ to complete in parallel to the operation of a (3,2) counter, where Δ_{OR_4} and $\Delta_{(3,2)}$ are the delays of the corresponding cells.
To deal with the various $(\{m, i, j\}, 3)$ units that may be needed in different implementations of saturating counters we have synthesized all the possible $(\{m, i, j\}, 3)$ units for $2 \leq i \leq 16$ and $2 \leq j \leq 7$. To reduce the number of different versions needed we have separated the part that

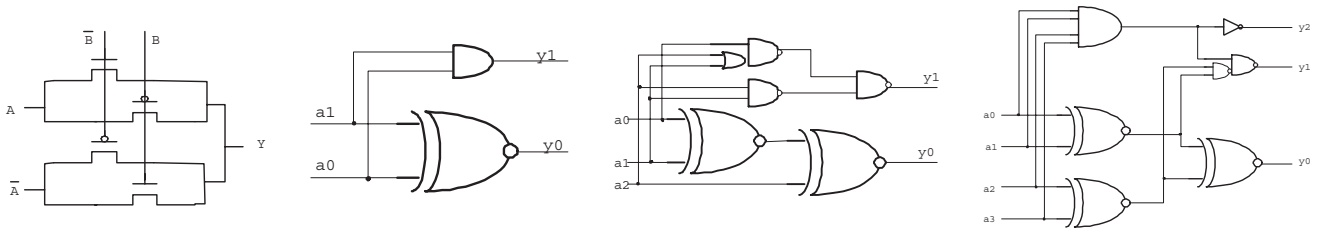

Figure 2. (a) XOR structure. (b)-(d) The implementation of the (2,2), (3,2) and (4,3) basic counters.

cell	area (μm^2)	delay (ns)	relative area	relative delay
OR_4	20.0	0.38	0.9	0.9
(2,2)	34.5	0.42	1.5	1.0
(3,2)	71.5	0.97	3.1	2.3
(4;2)	136.0	1.47	5.9	3.5
(4,3)	106.5	1.10	4.6	2.6

Table 5. The area and delay of the basic cells.

	j=2	j=3	j=4	j=5	j=6	j=7
i=2	10/1.0	21/1.1	29/2.4	33/2.7	42/2.8	57/3.3
i=3	16/1.8	23/1.8	37/2.6	45/2.9	47/2.8	57/3.2
i=4	22/2.6	22/2.6	44/2.7	47/3.1	47/3.0	57/3.2
i=5	32/2.8	36/2.8	43/3.0	53/3.1	62/3.2	76/3.6
i=6	28/2.7	42/2.6	48/2.7	57/3.1	60/3.5	74/3.4
i=7	40/3.0	42/3.0	49/3.1	68/3.1	61/3.5	77/3.6
i=8	40/3.2	42/3.1	50/3.4	68/4.1	68/3.7	81/3.6
i=9	55/3.5	55/3.6	61/3.9	83/4.0	78/3.9	91/3.8
i=10	54/3.8	53/4.2	77/3.7	80/4.2	83/4.2	95/4.2
i=11	57/4.1	67/3.5	72/3.8	85/3.6	91/3.6	103/3.6
i=12	60/3.9	79/3.5	87/3.9	89/3.9	101/4.0	109/4.0
i=13	82/4.2	82/3.6	102/4.1	97/4.1	108/4.6	108/4.2
i=14	73/4.3	85/4.3	97/4.4	103/4.4	112/4.4	111/4.5
i=15	85/3.9	84/4.0	100/4.8	108/4.8	112/4.4	120/4.5
i=16	75/4.6	88/4.3	103/4.9	116/4.9	115/4.8	120/4.8

Table 6. The area/delay for $(\{m, i, j\}, 3)$ units.

handles the m most significant bits from the rest of the circuit. Only OR_4 gates are used in this part and therefore, its contribution to the area of the circuit is approximately $\lceil (m/4) \rceil \cdot A_{OR_4}$ where A_{OR_4} is the area of an OR_4 gate. Furthermore, this part is not on the critical path of the complete circuit and thus need not be taken into account when calculating the delay of the entire circuit. The results of the synthesis are shown in Table 6. These implementations, unlike those summarized in Table 5, were obtained with the objective of minimum delay rather than minimum area. The basic cells in Table 5 are used repeatedly in the design of the saturating counter while only a single $(\{m, i, j\}, 3)$ unit is used. Thus, to achieve a reasonable trade-off between area and delay we prefer to use minimum area basic cells but minimum delay $(\{m, i, j\}, 3)$ unit.

In [4] we have shown that the use of an $(\{m, i, j\}, 3)$ yields a better solution for the implementations of a saturating counter which were considered there. In all our experiments with the synthesized cells described above, we have observed the same situation. Consequently, we show in the figures below the area and delay results only for counters which do use an $(\{m, i, j\}, 3)$ unit.

Figure 3 compares the delay of an [n,7] saturating counter implemented in four different ways: using (3,2) and (2,2) counters, allowing the use of (4;2) compressors as well, using (4,3), (3,2) and (2,2) counters and finally, allowing all the above types of counters.

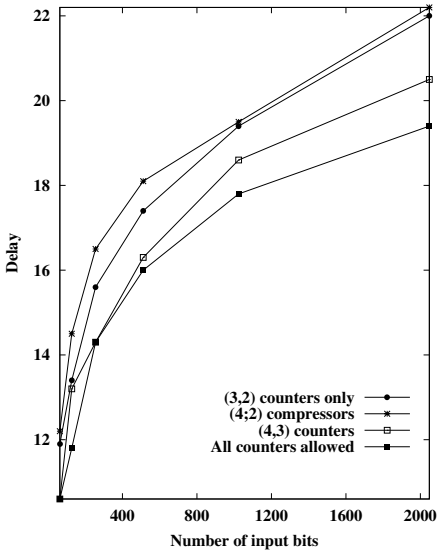

Figure 3. Delay comparison of four different implementations of an [n,7] saturating counter (all using a suitable $(\{m, i, j\}, 3)$ unit).

The results indicate that contrary to the conclusion drawn in [4], the use of (4;2) compressors does not necessarily reduce the delay of saturating counters and for the selected synthesized cells their use increases the total delay. Using (4,3) counters instead of (4;2) compressors, in addition to (3,2) and (2,2) counters, leads to a lower delay. As expected, allowing all three types of counters and the (4;2) compressor yields the best solution. However, due to the higher compression rate of the (4;2) compressor, preferring these compressors over the other basic counters yields the

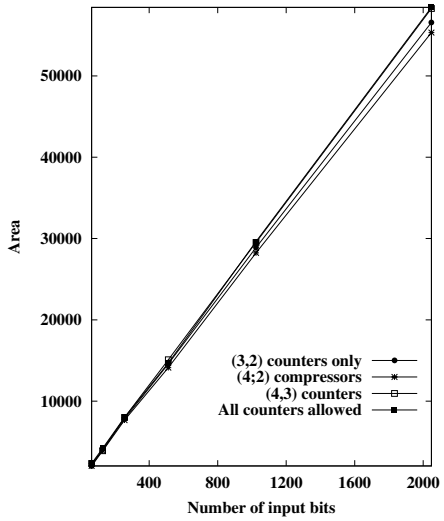

Figure 4. Area comparison of four different implementations of an [n,7] saturating counter (all using a suitable $(\{m,i,j\},3)$ unit).

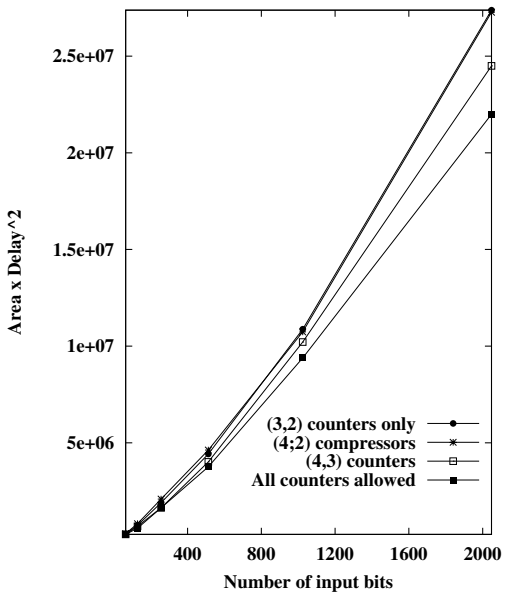

Figure 5. *Area* $\times$ *Delay*^{2} **comparison of four different implementations of an [n,7] saturating counter (all using a suitable $(\{m,i,j\},3)$ unit).**

lowest overall area of the saturating counter, see Figure 4. To make a decision regarding the trade-off between area and delay, a measure like *Area* $\times$ *Delay*^{2} may be used, and Figure 5 shows that allowing the use of the above basic cells provides the best solution by this standard.

We then analyzed the behavior of different designs of saturating counters as a function of the threshold T. As the threshold T approaches the number of input bits n, the saturating counter approaches the standard parallel counter and we therefore expect the benefits of using an $(\{m,i,j\},3)$ unit to diminish. To this end, we show in Figure 6 the delay of a saturating counter as a function of the threshold T for the above mentioned three basic designs which use an $(\{m,i,j\},3)$ and a design which uses (4,3) counters but does not include an $(\{m,i,j\},3)$ unit. The results confirm our expectations and also show that for high values of T (including $T = n$), (4,3) counters no longer outperform (4;2) compressors.

The differences among the four different implementation analyzed in Figure 6 are even more noticeable when examining the area of the saturating counter as shown in Figure 7. The use of (4;2) compressors yields the smallest area for a wide range of values of the threshold T. Figure 8 shows the *Area* $\times$ *Delay*^{2} of the above four implementations. For low values of T ($T \leq 64$), (4,3) counters are preferable while for higher values of T (4;2) compressors yield a better design.

So far we have restricted ourselves to a single implementation of the basic cells and used the minimum area - maximum delay one. The rational behind this decision was that these basic cells are used repeatedly throughout the design and a lower area will lead to a smaller value of the *Area* $\times$ *Delay*^{2} metric. However, there are stages in the saturating counter which include very few basic counters and for these stages a faster implementation (with a larger area) of the basic counter may lead to an overall better design. To study the above we have experimented with different im-

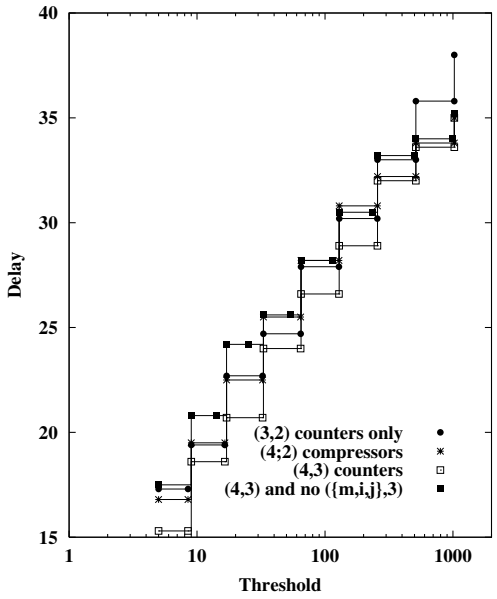

Figure 6. Delay of four different designs of a saturating counter as a function of the threshold.

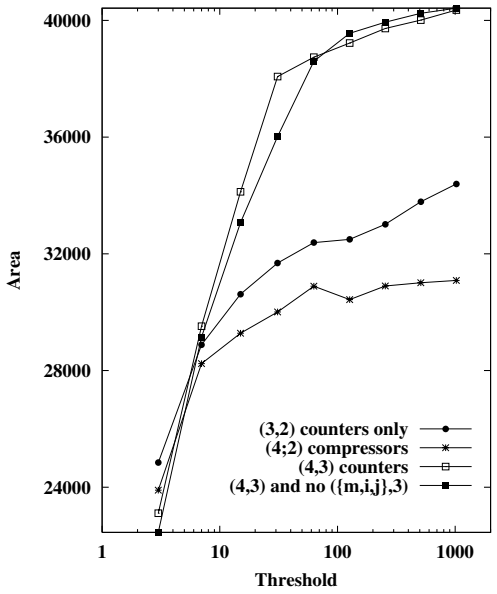

Figure 7. Area of the four different designs as a function of the threshold.

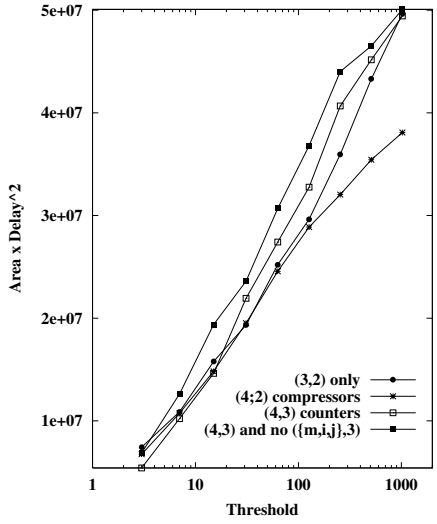

Figure 8. $Area \times Delay^2$ **of the four different designs as a function of the threshold.**

plementations of the (4,3) counter since it proved to yield better solutions for low values of the threshold. We have synthesized five variations of the (4,3) counter with different delay and area, shown in Figure 9. Out of these five, the one with the smallest area has been used in the previous designs. For this implementation we get the top curve in Figure 10 which depicts the delay of the [n,7] saturating counter that uses (4,3) counters. If instead we use the smallest delay implementation of a (4,3) counter out of the five in Figure 9, we obtain the bottom curve in Figure 10, i.e., a considerably lower overall delay. The middle curve in Figure 10 is obtained by selectively using all five implementations of the (4,3) counter. In the top stage which includes the largest number of (4,3) counters we use the minimum area implementation. As we move down to the second stage we use instead an implementation with a slightly larger area but smaller delay and so on. The decision which implementation to use in each stage is done so as to minimize the $Area \times Delay^2$ of the saturating counter. The particular selection of (4,3) counter implementations for the different stages of the saturating counter that corresponds to the middle curve in Figure 10 was determined after a limited number of trials. Thus, the optimality of the presented solution is not guaranteed but we believe that the presented solution is reasonably close to the optimal one. The corresponding curves for the area and the $Area \times Delay^2$ for the above three designs are shown in Figure 11 and 12, respectively. As expected, the curve in Figure 11 for the minimum area implementation of the (4,3) counter is the bottom one and in Figure 12, the curve for the variable delay/area implementation of the (4,3) counter is the bottom one.

4. Conclusion

Saturating counters have been defined and several design alternatives using various basic counters (traditional and non-traditional) have been presented and evaluated. For low values of the threshold, (4,3) counters with variable delay/area implementation plus an $(\{m, i, j\}, 3)$ unit proved to yield the best implementation when using the $Area \times Delay^2$ metric. For high values of the threshold when the saturating counter approached a conventional parallel counter, (4;2) compressors without the $(\{m, i, j\}, 3)$ unit yield a better design.

We have also experimented with several other non-

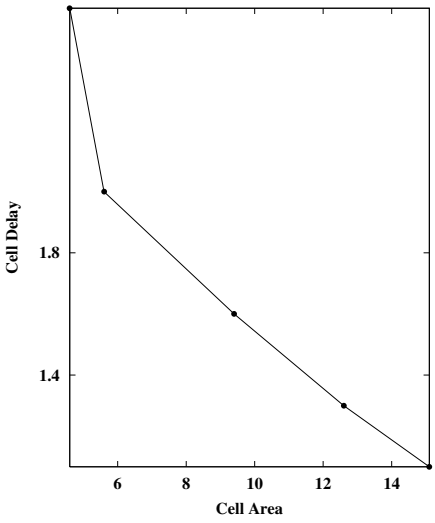

Figure 9. Five different implementations of a (4,3) counter.

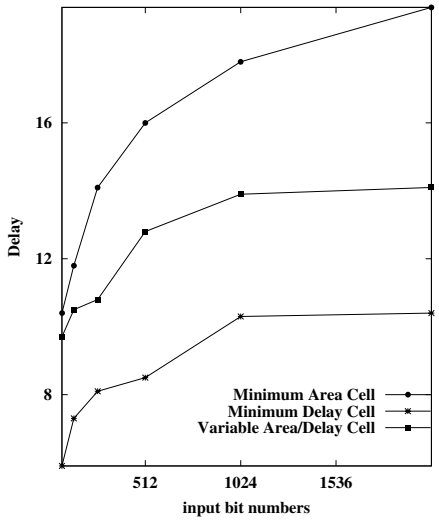

Figure 10. Delay of the [n,7] saturating counter using different implementations of (4,3) counters.

traditional basic counters like (5,3) and (8,4). Although these did further reduce the delay of the saturating counter, the resulting area overhead was too high. Other non-traditional counters and some combinations of these may, however, prove to yield better implementations of saturating counters.

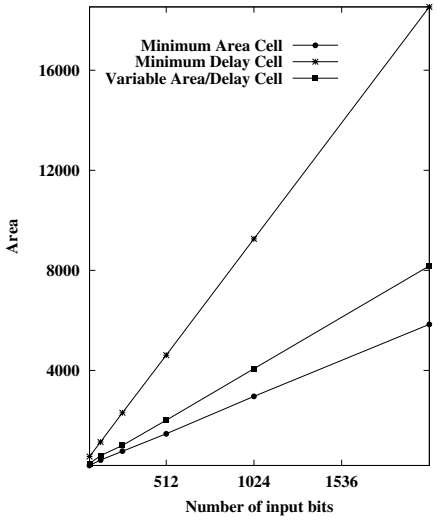

Figure 11. Area of the [n,7] saturating counter using different implementations of (4,3) counters.

References

[1] R. F. Jones and E. E. Swartzlander, "Parallel Counter Implementation," *Journal of VLSI Signal Processing,* pp. 223-232, 1994.

[2] E. E. Swartzlander, "Parallel Counters," *IEEE Trans. on Computers*, Vol. C-22, pp. 1021-1024, 1973.

[3] I. Koren, *Computer Arithmetic Algorithms*, 2nd edition, A K Peters, Natick, MA, 2002.

[4] I. Koren, Y. Koren and B. Oomman, "Saturating counters: Application and Design Alternatives," *Proc. of the 16th IEEE Symp. on Computer Arithmetic*, pp. 228-235, June 2003.

[5] N. Ohkubo, M. Suzuki, *et. al.*, "A 4.4-ns CMOS 54 $\times$ 54-b Multiplier Using Pass-Transistor Multiplexor," *IEEE Journal of Solid-State Circuits*, vol.30, pp. 251–256, Mar. 1995.

[6] http://www.ecs.umass.edu/ece/koren/arith/simulator/SatCount/

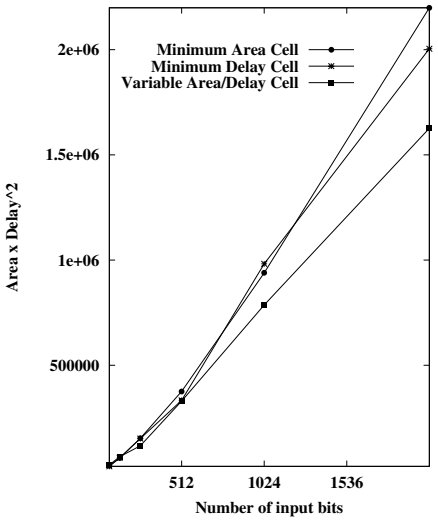

Figure 12. $Area \times Delay^2$ **of the [n,7] saturating counter using different implementations of (4,3) counters.**