Synthesis and Competitive Analysis of Multiple Inputs Parallel Adders

Dimitar S. Tyanev, Stefka I. Popova, Aleksandar I. Ivanov, Dragomir V. Yanev

Abstract: *This paper submit the idea for parallel at time adding of more than two integer numbers with schemes, known as concentrators. A logic structure of adder with multiple inputs, based on three-input adder, is offered. Theorem for the highest length of the sum is proved in conditions of arbitrary numbers. On the ground of this theorem there are analytical estimations of implementation costs and switching time. Comparative analysis of the competitive schemes is shown. The conclusions proof acceptability of both researched idea and offered scheme. The experiments with the schemes were made with the help of Xilinx tools and FPGA-family Spartan II.*

1. Problem formulation

There are many applications that generate the problem of computing the sum like:

$$y = \sum_{i=1}^{r} x_i \tag{1}$$

where x_i, $i = \overline{1,r}$ usually are n-bits numbers form the same type and format.

As subtask, the sum (1) appears in the vector-matrix computations, in the problems of mathematical statistic, the problems of digital processing of images and many more. As it showed in the quoted literature [1÷12 and other], the idea about parallel adding is known in principle. This idea is present in the contemporary analyses, searching for fast machine one part time and/or conveyer decisions. There are different suggestions about the adding, but predominant is the use of binary schemes with three inputs and so-called saved carry (carry save adders). At the same time there is lack of analytical and competitive estimations, as well as the reasons for the choice. In [7] there is method for analysis of transitional processes in similar schemes.

In this paper we offer another machine implementation, as well as our examination of it, which is shown consecutively.

2. Consecutive addition

The classic two-seated arithmetic operation addition

$$y = x_1 + x_2 \tag{2}$$

corresponds to (1) and is realized by known logical schemes of n-bit binary adder. It is known [5], that the most inexpensive and the fastest at the same time binary combinational adder is the adder with accelerated consecutive carry, which we choose for base, and it has switching time

$$t_{\Sigma} = n.\tau \quad , \tag{3}$$

where τ is the switching time of full one-bit binary combinational adder.

So the task for computing of (1) can be solved with the help of next logical structure:

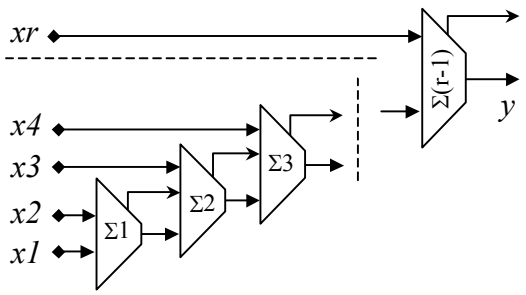

Figure1. Consecutive addition

Although at the structure above all the elements of the sum are given simultaneously, the result y is received by consecutive adding of the particular mediate sums Σ_i with the current operand x_i. However, every single of these mediate sums could be prolonged with maximum one bit from the left because of the arising carry. So, if the adder Σ_1 has n-bits, then the adder Σ_2 got to have $(n+1)$-bits. Further, by analogy follows adder Σ_3 – $(n+2)$-bits, adder Σ_4 – $(n+3)$-bits and so on. The last adder Σ(r-1) got to have $(n+r-2)$ bits. It is expected, that full adding in this adder will be realized only in the junior n-bits, where the last addends x_i are given. Full addition in the senior $(r-2)$ bits is impossible. Only distribution and adding with likely carry from the $(n-1)$-bit is possible. As we consider, that the length of the addends x_i is one and the same, the possibility about prolonging of the mediate sum decreases as result of the separate levels. This means that determined above length of the last adder is unreal. It is the same for the length of the adders from lower levels. But there is no doubt that the highest possible length of final sum depends from the count of addends r. This non-conformity leads to the following theorem:

Theorem:

The highest possible length of the sum of r n-bit integer numbers is equal to

$$n+\lfloor log_2(r)\rfloor+1 \text{ [b].} \tag{4}$$

Proof:

Highest possible sum of r numbers from the same type is received when these numbers has the highest possible for their length value $X = 2^n - 1$. In this case the sum (1) of these numbers from the same type could be described as follows:

$$\sum_{i=1}^{r} x_i = r.X \quad . \tag{5}$$

The number with highest value X actually is binary polynomial from $(n-1)$ row. At the same time for the multiplicand r in (5) we can assert, that as binary number is in the range:

$$2^q < r < 2^{q+1} , \tag{6}$$

where the row of its binary polynomial is q, i.e. $r = P_r^{(q)}$. So, the unknown quantity q can be found from the inequality above, in the conditions of certain excess, after operation logarithm:

$$q = \lfloor \log_2(r) \rfloor \quad . \tag{7}$$

It is known, that the quantity of digits in a number is bigger with one from the row of its polynomial [5]. So the number r in binary system will have $(q+1)$ digits. As a result, the highest possible length, required for registration of the highest sum from r elements, is equal to:

$$n+(q+1) = n+(\lfloor log_2(r)\rfloor+1) \quad \text{[b]} , \tag{8}$$

Q.E.D.

In the special case, when $r = 2^q$, the required length for the sum is $(n+q)$.

Proven theorem provides true necessary and sufficient length for registration of every sum with rate frequency r and description like (1). The value of deduced extension $(q+1)$ [b] for result's field is at the same time minimum required. Comparing theorem's result with defined length of final adder in structure at Figure 1 – (n+r-2) [b], we will prove, that last is bigger than necessary and is unreal, as was mentioned above.

This statement we can describe with following ratio:

$$n+(q+1) \quad < \quad n+(r-2) \quad . \tag{9}$$

After replacing of q and transforming of expression, the equivalent ration is

$$\lfloor \log_2(r) \rfloor \quad < \quad r-3 \quad . \tag{10}$$

Then with anti-logarithm of the two sides of the ratio:

$$r \quad < \quad 2^{r-3} \quad . \tag{11}$$

The last is true for every $r>6$.

The logical structure from Figure 1 can be optimized to correspond to required and enough length (4) in connection with proven ratio. For that purpose, if the length of basic adder is determined by (4), then on the strength of the proven theorem it has to be applied for every single lower level of the structure. This means for the first *(r-1)* levels, then for the first *(r-2)* levels and so on. Finally in the structure there will be particular groups from adders with one and the same length, which definitely will not generate senior carry.

The sum of different level's implementation outlays represents total cost for machine realization. The outlays for implementing adders at levels depends from their current length, which can be represent like

$$L_k = n + \lfloor \log_2(k) \rfloor, \quad k = \overline{1, r-1}. \tag{12}$$

where k is the number of the level, where the serial adder belongs. So machine cost for implementation of the structure from Figure 1 can be estimated through total count of binary adders by next sum:

$$Q = (r-1).n + \sum_{k=1}^{r-1} \lfloor \log_2(k) \rfloor \ . \tag{13}$$

Second estimation, which should be taken into consideration, is logical structure's switching time. Because of nature of the process, realized by the scheme, switching time can be defined from how far carry is distributed. Since carry is distributing across length of every adder and through all levels to the final, then estimation is

$$t_\Sigma = (n + 2.r - 3).\tau \ . \tag{14}$$

Notice that this estimation (as highest) cannot be reached for every r, because the carry is function of current addends.

3. Parallel addition

Analyzing structure from Figure 1 we can mention, that the only pure parallel addition is accomplished between numbers *x1* and *x2* in adder Σ_1. All other numbers – *x3, x4, x5, ..., xr* consecutively heap up in the final sum. For the sake of faster operation of the scheme, we suggest parallel organization to be propagating through all possible pairs of addends, which lead us to Figure 2.

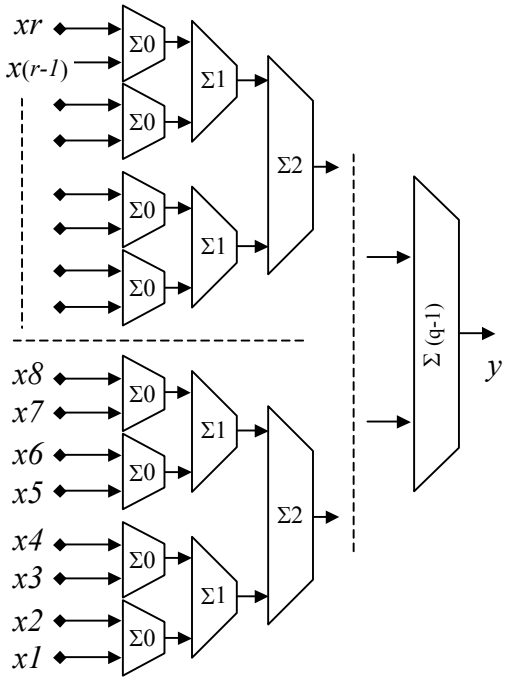

Figure 2. Parallel-consecutive addition

As it shown, *r/2* two-input *n*-bits adders Σ0 are arranged at first row. At the next row the count of adders decreasing two times again, but adders Σ1 have (*n*+1)-bits. So is certain force to make

presented consecutive connection, which forms pyramid-like model of the scheme. The count of levels in this pyramid is q, where last adder Σ_{q-1} has $(n+q)$-bits. In conjunction with the senior carry those adder receives $(n+q+1)$-bits result y.

The machine implementation cost for the structure from Figure 2 can be estimated analogically:

$$Q = n.N_0 + (n+1).N_1 + (n+2).N_2 + ... + (n+q-1).N_{q-1} \,., \tag{15}$$

where the count of the binary adders at particular levels changes as follows

$$r_0 = r \;; \quad N_0 = \left\lfloor \frac{r_0}{2} \right\rfloor \;; \quad N_1 = \left\lfloor \frac{N_0 + (r - 2.N_0)}{2} \right\rfloor = \left\lfloor \frac{r_1}{2} \right\rfloor \;; .... ; \quad N_{q-1} = 1 \;.$$

In case of odd number addends at one level, remaining element, which cannot form pair addends, is directly sent to the next level, where in conjunction with received intermediate sums takes part in the next pair's addends. That's why we will control the cyclic accumulation by condition, depending of the parameter k – the count of levels in pyramid-like structure.

In accordance with the upper row, generally the number of the binary adders at the pyramid's level k will be as follows:

$$N_k = \left\lfloor \frac{N_{k-1} + (r_{k-1} - 2.N_{k-1})}{2} \right\rfloor = \left\lfloor \frac{r_k}{2} \right\rfloor \;; \quad r_k = N_{k-1} + (r_{k-1} - 2.N_{k-1}) \,. \tag{16}$$

Computations (16) will end when current number of the addends is equal to 1, i.e. when $r_k = 1$.

Then the sum of 1-bit binary adders (15) will be finally represented through (16) like:

$$Q = \sum_{k=0} \left\{ (n+k).\left\lfloor \frac{r_k}{2} \right\rfloor \right\} \;. \tag{17}$$

Switching time of the structure from Figure 2 is again estimated by how long is distributed carry for obtaining total sum in condition of adders with accelerated consecutive carry. As we consider, that the adders from every particular level works parallel, switching time of the pyramid-like structure can be estimated as

$$t_\Sigma = (n+q-1).\tau \;. \tag{18}$$

The sum reflects the length of adders in first level, which pass on received digits to the next level right after their obtaining, where addition begins. Final sums at the output of the second level will be late regarding to the initial moment, because of their additional senior bit. This switching refers to all other levels.

4. Parallel addition by 3-input adders

In contrast to idea for unfinished adding of three addends (3:2 CSA), discussed in [1, 2 and others], [7] offers and estimates finalized scheme of an adder with 3-input concentrator. Achieved scheme (Figure 3) is of type 3:1 CSA. Because this scheme has one output, its usage reduces the count of levels. On the other side, it reduces the count of fixing registers in case of structures with conveyer organization.

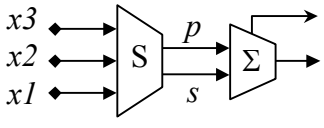

Figure 3. 3:1 CSA

In this structure concentrator S is set of n complete 1-bit binary adders, which operate in parallel mode. Latest scheme Σ is complete adder, which adds intermediate sums s_i with relevant carry p_{i-1}, $i = \overline{1,n-1}$.

Technical estimations of the structure for n-bits numbers, shown at Figure 3, are represented by next formulas. Cost of machine implementation is estimated by count of the 1-bit binary adders:

$$Q = n + n = 2.n \ , \tag{19}$$

and switching time – by the switching time of the 1-bit binary adder:

$$t_{\Sigma} = \tau + n.\tau = (n+1).\tau \ . \tag{20}$$

For initial task (1) instead of parallel addition of two numbers, what was suggested in section 3, it will be structured upon groups of 3 numbers. Using such approach in random situation – *r*-times addition of *n*-bits numbers, guarantees minimal count of levels in the pyramid-like structure. Generalized estimations about such structures are achieved through to an example - 8 times addition of 8-bits numbers (Figure 4).

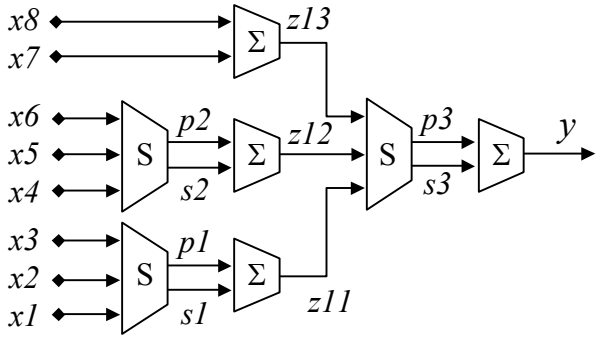

Figure 4. Adding scheme with 4 inputs

At the first level of the structure, the count of the groups *N* has 3 addends, in which separates the initial multitude of r addends, can be defined as

$$N_1 = \left\lfloor \frac{r_1}{3} \right\rfloor \ , \qquad \text{where} \ \ r_1 = r \ . \tag{21}$$

If the residual $Rem = (r_1)_{\text{mod}3} = 1$, there is one unused addend left. This addend is sent to the next level of the pyramid, where forms triad with obtained from the first level sums $z1_i$. In the case $Rem = (r_1)_{\text{mod}3} = 2$, there are two unused addends left. Then additional full binary adder is applied at the current level, what is the case with addends *x7* and *x8* in Figure 4.

So the estimation of the machine cost Q_1 for implementing of this level, represented like number of 1-bit binary adders, is:

$$\textbf{\textit{if}} \quad Rem_1 = 2 \quad \textbf{\textit{then}} \quad Q_1 = 2.n.N_1 + n = n.(2.N_1 + 1)$$
$$\textbf{\textit{else}} \quad Q_1 = 2.n.N_1 \ .$$

The arrangement and implementation of the following second level is similar. The number of 3 addends groups N_2 could be represent like:

$N_2 = \left\lfloor \frac{r_2}{3} \right\rfloor$, where the general number r_2 is defined by the rule:

$$\textbf{\textit{if}} \quad Rem_1 \neq 0 \quad \textbf{\textit{then}} \quad r_2 = N_1 + 1 \ ; \quad \textbf{\textit{else}} \quad r_2 = N_1 \ .$$

The arrangement and implementation of the following second level is similar. The number of 3 addends groups N_2 could be represent like:

$$\textbf{\textit{if}} \quad Rem_2 = 2 \quad \textbf{\textit{then}} \quad Q_2 = 2.(n+1).N_2 + (n+1) \ ;$$
$$\textbf{\textit{else}} \quad Q_2 = 2.(n+1).N_2 \ .$$

To synthesize arbitrary level, upper rules can be generalized as follows:

$$N_{k+1} = \left\lfloor \frac{r_{k+1}}{3} \right\rfloor \ , \tag{22}$$

where the general number addends r_{k+1} is defined by the rule

$$\textbf{\textit{if}} \quad Rem_k \neq 0 \quad \textbf{\textit{then}} \quad r_{k+1} = N_k + 1 \quad \textbf{\textit{else}} \quad r_{k+1} = N_k \ . \tag{23}$$

So the estimation for total machine cost Q_{k+1} will be:

$$\textbf{\textit{if}} \quad Rem_{k+1} = 2 \quad \textbf{\textit{then}} \quad Q_{k+1} = 2.[n + 2.(k-1)].N_{k+1} + [n + 2.(k-1)]$$
$$\textbf{\textit{else}} \quad Q_{k+1} = 2.[n + 2.(k-1)].N_{k+1} \quad . \tag{24}$$

If current $r_k > 3$, the synthesis of the structure continues in the same way. Remaining two cases leads to the last level in the pyramid, which is synthesized as follows:

1. If $r_k = 3$, the pyramid ends with the structure from Figure 3, like the example from Figure 4. In this case machine cost for the last level is $Q_k = 2.[n + 2.(k-2)]$;
2. If $r_k < 3$, the pyramid ends only with full binary adder. In this case machine cost is $Q_k = n + 2.(k-2)$.

The relation $r_k > 3$ is actually the condition about continuing or ending general rules for synthesizing the pyramid. At the end total machine cost are the sum of implementation outlays of particular levels:

$$Q = Q_1 + Q_2 + ... + Q_k \quad . \tag{25}$$

The switching time of the structure at Figure 4 with arbitrary number of addends is estimated again by the length of consecutively distributed carry when forming total result and in condition of adders, constructed as described. Considering, that 3-input adders at particular levels works parallel, we estimate the switching time of the pyramid-like structure as:

$$t_{\Sigma} = \tau + n.\tau + (k-1).\tau = (n+k).\tau \quad . \tag{26}$$

The logic of this switching is analogical to the described for structure at Figure 2.

5. Theoretical conclusions

Comparative estimations of the machine costs, necessary for implementation of the three structures, which were considered, explained by the formulas (13), (17) and (25), are represented in Tables 1, 2 and 3 respectively, and also in Figure 5. The results are for the operands with length n=32[b].

Table 1 Consecutive addition

r=	8	16	24	32	40	48	56	64	72	128	256
Q_k=	234	514	802	1090	1386	1682	1978	2274	2578	4700	9685

Table 2 Parallel addition

r=	8	16	24	32	40	48	56	64	72	128	256
Q_k=	228	491	755	1018	1283	1546	1810	2073	2339	4184	8407

Table 3 Addition by 3-input schemes

r=	8	16	24	32	40	48	56	64	72	128	256
Q_k=	228	492	754	1018	1282	1544	1808	2073	2339	4184	8410

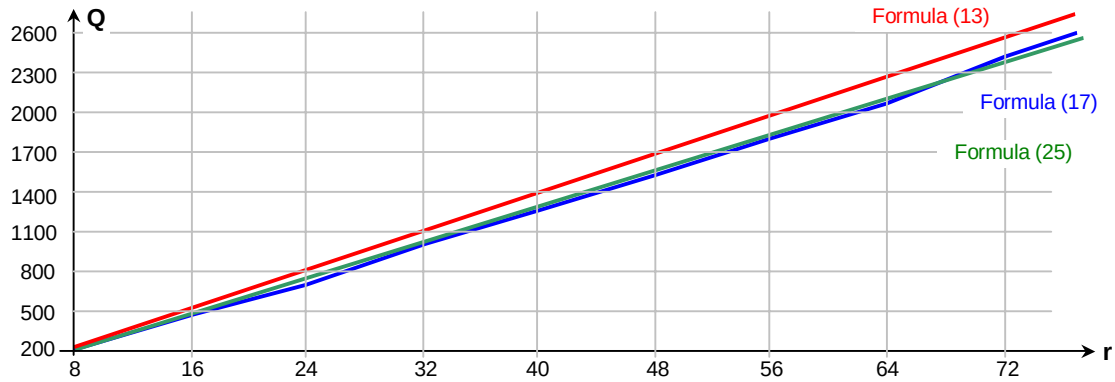

Figure 5. Machine costs Q depending from count of addends r

The most important conclusions, which can be made from obtained results, are:

1. The machine costs in case of parallel organization are 10% lower;
2. 2- and 3-input schemes for parallel addition leads to identical machine costs;

The comparative estimations of the operation of explained structures, on the basis of the time for most delay through switching, determined by the formulas (14), (18) and (26), are represented in the Table 4, 5 and 6 respectively, as it Figure 6.

Table 4 Consecutive addition

r=	8	16	24	32	40	48	56	64	72	128	256
t_Σ=	45.τ	61.τ	77.τ	93.τ	109.τ	125.τ	141.τ	157.τ	173.τ	285.τ	541.τ

Table 5 Parallel addition

r=	8	16	24	32	40	48	56	64	72	128	256	1024	2048
t_Σ=	34.τ	35.τ	35.τ	36.τ	36.τ	36.τ	36.τ	37.τ	37.τ	38.τ	39.τ	41.τ	51.τ

Table 6 Addition by 3-input schemes

r=	8	16	24	32	40	48	56	64	72	128	256	1024	2048
t_Σ=	34.τ	35.τ	35.τ	36.τ	36.τ	36.τ	36.τ	36.τ	36.τ	37.τ	38.τ	39.τ	39.τ

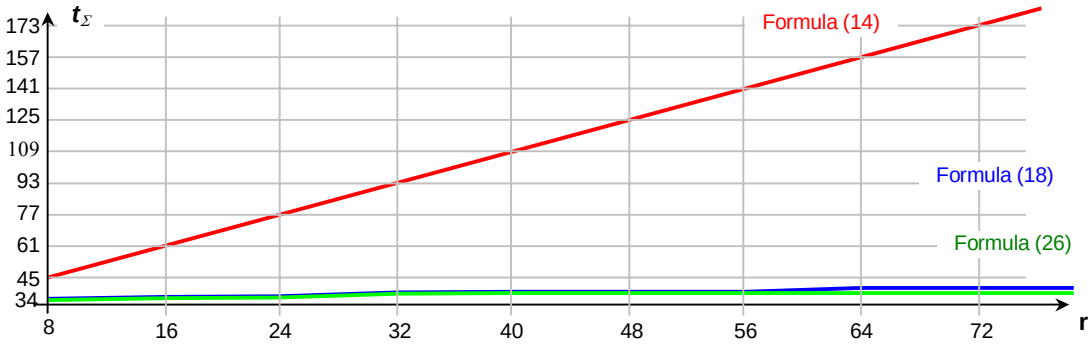

Figure 6. Switching time t_Σ depending form the count of addends r

The main conclusions from the results are:

1. Significant superiority of the parallel organization over consecutive. This is because of considerable smaller number levels in the pyramid-like organization;
2. 2- and 3-input schemes for parallel addition have slowly increasing difference in favor of the second scheme. The absolute difference between them reaches 12 relative time units for 2048-times adding. But at the beginning of relations (for example to 128-times addition) they can be considered as analogical (with accuracy to 1).

6. Experimental results

In conformation of achieved theoretical estimations about switching time, presented in tables 5 and 6, an experimental research and comparing of the logical structures form Figure 2 and Figure 4 was organized. The case of 8 time addition of 8-bit binary numbers was realized. As a merit of the scheme from Figure 4 it is necessary to underline, that it can be loaded up with one more number (in difference from the structure at Figure 2). It also can add 9 numbers with almost the same machine resources and with the same switching time, which is not possible for the scheme at Figure 2. The implementation of the schemes was made in ISE environment of Xilinx – WEBPack. Under these conditions estimations of switching time in worst case, according to formulas (18) and (26), are the same: $t_\Sigma = (8+3-1).\tau$; $t_\Sigma = (8+2).\tau$. As we consider the logical scheme of 1-bit binary adder, for whom the estimation τ for even bits is proportional to 2 logical elements, and for odd bits – to 3 logical elements, we can assert, that the common switching time of the schemes is in the range of [20÷30] relative time units.

Both of the upper theoretical conclusions are completely confirmed through to experiments, which can be seen from the diagrams at Figure 7 and Figure 8.

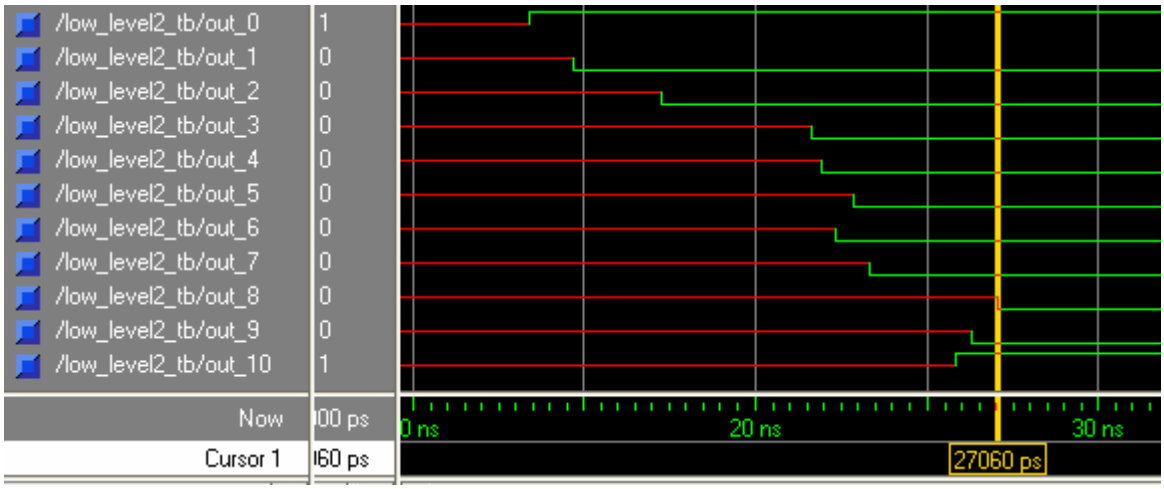

Figure 7 Delays in the scheme with 2-input binary adders

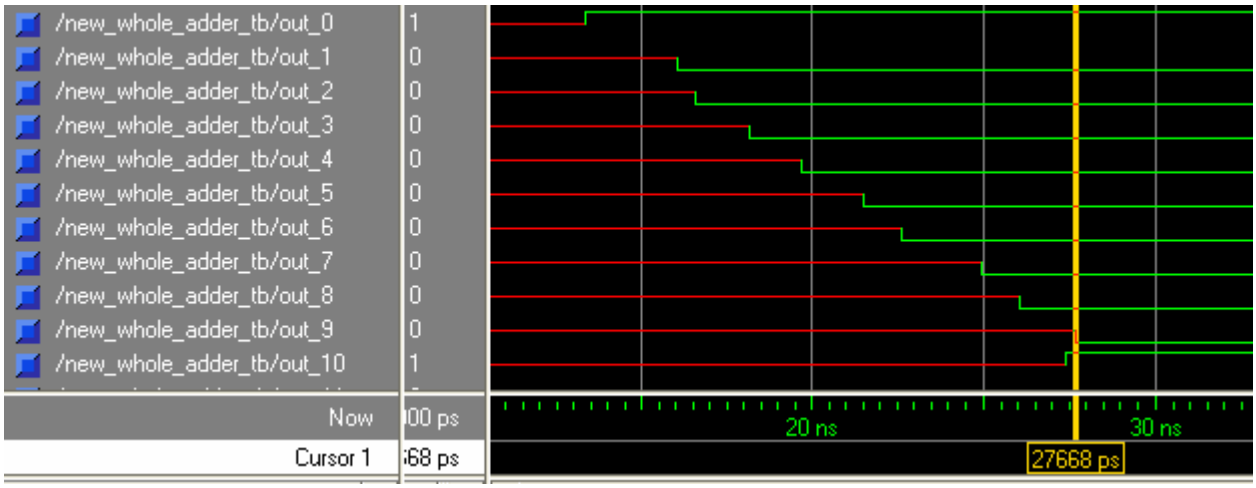

Figure 8 Delays in the scheme with 3-input binary adders

As it showed, the delays of both schemes can be assumed as equal, so as their real values which are in range of 27 relative time units. Presented diagrams are achieved through addition of the following 8 numbers:

(3+255) + (255+0) + (255+0) + (128+129) = (255+255+3) + (0+0+255) + (128+129) = 1025.

7. References

[1]. ASIC Design for Signal Processing - Carry Save Arithmetic: http://www.geoffknagge.com/fyp/ carrysave.shtml

[2]. Loh, *Carry Save Addition*, Processor Design, spring 2005, http://www3.cc.gatech.edu/classes/AY2005/cs3220_spring/csa-notes.pdf.

[3]. *Digital Computer Arithmetic Datapath Design Using Verilog Hdl*, James E. Stine, 2004. http://books.google.com/books?vid=ISBN1402077106&id=Jc3cz5dvIxYC&pg=RA1-PA60&lpg=RA1-PA60&ots=K-meqDviLN&dq=design+carry+save+adder&sig=1fINOk4NXsgKFZIOIHzAcmePPIc#PRA1-PA179,M1

[4] B. Parhami, *"Computer Arithmetic" - part Carry Save Arithmetic*, Oxford Press, 2000, pp.131.

[5]. Тянев Д. С., *Организация на компютъра (цифрова аритметика),* ISBN 954-20-0258-0, ТУ - Варна, 2004 год.

[6]. Карцев М. А., *Арифметика цифровых машин,* Издательство “Наука”, Москва, 1969.

[7]. Карцев М. А., Брик В. А., *Вычислительные системы и синхронная арифметика*, Издательство “Радио и связь”, 1981.
[8]. *Специализированные процессоры для высокопроизводительной обработки данных*, Новосибирск, Издательство “Наука”, 1998.
[9]. Фет И., *Специализированные однородные структуры. Цифровые компрессоры,* АН Институт Математики, Препринт №27, 1988.
[10]. Kiefer G., Waker A., Thumm A., *Rechnerorganisation,* Institut fur Informatik, Stuttgart, 2001.
[11]. Schiller, Jochen, *Rechnerstrukturen,* Freie Universitat, Berlin, 2003.
[12]. T. Kim, W. Jao, and S. Tjiang, "*Circuit Optimization using Carry-Save-Adder Cells*", IEEE Trans. on CAD, Vol.17, No.10,1998.