Transformation of a 2-D VLSI Systolic Adder Circuit in 3-D Circuits Using Optical Interconnections

Dietmar Fey

Friedrich-Schiller-Universität Jena, Institut für Informatik
D- 07740 Jena, Germany

Abstract. An optimal systolic adder is mapped onto three different optoelectronic 3-D systems. It is shown that compared to the 2-D system the latency can be improved from $o(\sqrt{n})$ to $o(\sqrt[3]{n})$ and the period time from $o(\sqrt{n})$ to $o(1)$.

1 Introduction

The increasing availability of optoelectronic integrated circuits with a high number of optical I/O pads [1] and optical interconnection modules for chip-to-chip communication [2] makes a near term development of 3-D high-integrated circuits possible. In so called smart pixel systems it is intended to combine high-integrated electronic logic with optical interconnections. Current realisations of smart pixel systems use photodetector-emitter [3] or photodetector-modulator combinations [1], [4] to connect integrated circuits through space.

Using fast and high-dense optical interconnections can solve some of the major problems in current VLSI technology: limited pin numbers and loss of performance due to broadcast connections and long on-chip wires. In addition to this quantitative improvement based on a higher connection density, a redesign regarding adequately the freedom the third dimension offers further qualitative improvements. In the presented paper this is demonstrated for the example of a specific systolic algorithm that was evaluated as optimal for a 2-D VLSI design [5]. This architecture can be transformed to a 3-D architecture under different transformation techniques. It will be shown that the 3-D approaches offer more performance in terms of latency, period time and throughput as the 2-D solution.

In Sec. 2 the systolic algorithm and the corresponding 2-D array are described. Sec. 3 shows three transformation techniques we applied to this 2-D architecture to receive equivalent 3-D solutions consisting of stacked 2-D smart pixel arrays. For these different approaches a comparative performance evaluation is carried out in Sec. 4. Finally we conclude with a summary and an outlook.

2 Optimal Systolic Adder Algorithm for 2-D Circuits

The base point of our investigation is an optimal systolic adder presented by Kühnel [5]. Optimal means that the specified architecture reaches the lower bounds of typical VLSI complexity measures depending on the wordlength *n*. These measures are

- the needed *area* A(n),
- the *latency time* T(n), i.e. the time to carry out a single operation, and
- the *period time* P(n), i.e. the elapsed time between two successive operations.

The algorithm represents a compromise between the fast but hardware intensive carry-look-ahead-technique and the slow but hardware sparing ripple-carry-technique [6],[7], [8]. Besides, the n bits of the two addends A and B are separated in $\sqrt{n}$ blocks with $\sqrt{n}$ successive bit pairs in each block. Fig. 1 shows this for two operands A and B. The notation [i,j] indicates a set of integer $\{k \,/\, i \leq k \wedge k \leq j\}$.

	3 blocks with 3 bit pairs each		
$A = A_8A_7...A_1A_0$	A,B [0,2]	A,B [3,5]	A,B [6,8]
$B = B_8B_7...B_1B_0$	$A_0\ B_0$	$A_3\ B_3$	$A_6\ B_6$
	$A_1\ B_1$	$A_4\ B_4$	$A_7\ B_7$
	$A_2\ B_2$	$A_5\ B_5$	$A_8\ B_8$

Fig. 1: Separation of operand pair A, B with wordlength $n = 9$ in 3 partitions a 3 bit

Within each block $[q\sqrt{n}, (q+1)\sqrt{n}-1], q \in [0, \sqrt{n}-1]$ the ripple-carry-technique is applied. Among the blocks the carry-look-ahead-technique is used to determine the spreading of the carry bits. In each block G(enerate) bits $\mathrm{G}[q\sqrt{n}, (q+1)\sqrt{n}-1]$ and P(ropagate)-bits $\mathrm{P}[q\sqrt{n}, (q+1)\sqrt{n}-1]$ are calculated to indicate if a carry bit is generated in the block or if an incoming carry bit is propagated to the next block, respectively. These G- and P-bits are calculated starting from the first bit position in the block by using (1).

$$\begin{aligned} &G[q\sqrt{n},k] = (a_k \wedge b_k) \vee ((a_k \oplus b_k) \wedge P[q\sqrt{n},k-1]). \\ &P[q\sqrt{n},k] = (a_k \oplus b_k) \wedge P[q\sqrt{n},k-1]; k \in [q\sqrt{n},(q+1)\sqrt{n}-1],\, q \in (0,\sqrt{n}-1). \end{aligned} \tag{1}$$

A carry bit that is spreading over two and more blocks is called a *cumulative carry bit.* Cumulative carry bits are calculated with (2).

$$\begin{aligned} G[0,q\sqrt{n}-1] = G[(q-1)\sqrt{n},q\sqrt{n}-1] \vee \\ (P[(q-1)\sqrt{n},q\sqrt{n}-1] \wedge G[0,(q-1)\sqrt{n}-1)]) \ . \end{aligned} \tag{2}$$

With known cumulative carry bits it is possible to determine for each bit position if a carry bit arrives from left, i.e. G[0,i-1]=1 (3). Then we can determine the final sum bits s_i (4).

$$G[0,i-1] = G[q\sqrt{n},i-1] \vee (P[q\sqrt{n},i-1] \wedge G[0,q\sqrt{n}-1)]) \ . \tag{3}$$

$$s_i = a_i \oplus b_i \oplus G[0,i-1]\,; \quad i = q\sqrt{n}+m; m < \sqrt{n} \ . \tag{4}$$

The calculation of the sum bits and the cumulative block carries can be carried out in a pipelined mode with the systolic array shown in Fig. 2, which was proposed by Kühnel [5]. The processing cells A in the first row use the ripple-carry-procedure to calculate the intra-block carries (1). The processing cells B and C at the right edge are responsible for the generation of the cumulative block carries (3) and the sum bits (4). Cells marked with D delay their input bits for one clock cycle to synchronise the data flow. The bit combination (0,1), (0,1), (0,1), (0,0) allows to reset the necessary flip-flops in cell B at the beginning of a new calculation.

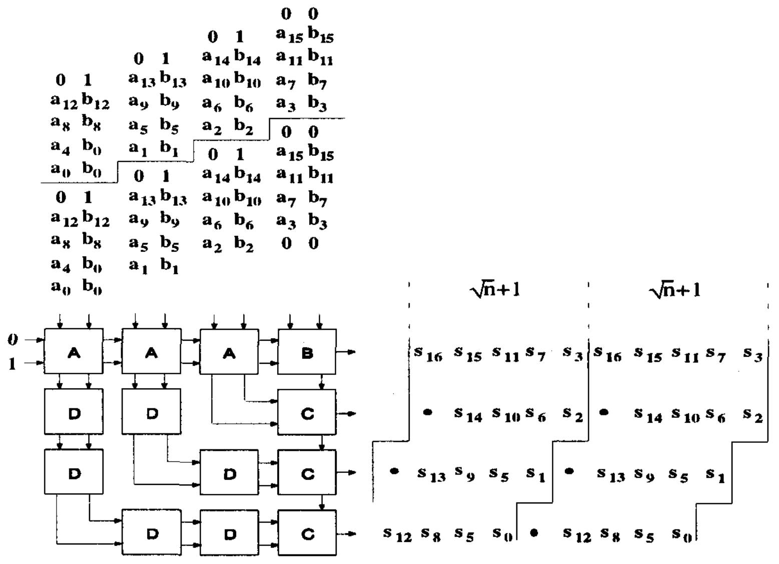


Fig. 2: 2-D array for a 16-bit systolic adder [5]

Always after $\sqrt{n}+1$ steps a new result is completely shifted out at the right edge of the array. Hence, the period time is $\sqrt{n}+1$. A single operation needs $3\sqrt{n}-3$ steps. This is exactly the number of time steps until the last bit s_{12} is shifted out after the corresponding input bit combination (a_{12},b_{12}) started to move towards the array. Hence, the latency is $3\sqrt{n}-3$.

Systolic architectures offer many advantages for VLSI circuit design as well as for the design of smart pixels due to the following features:

- The array contains only few different processing nodes making the design process for the young smart pixel technology easier and faster.
- The array and especially the interconnection structure is largely regular. This leads to a regular interconnection topology, which is easier to realise than an interconnection system consisting of completely irregular running „optical wires".
- Systolic arrays combine processor duplication and pipelining. This feature is ideal to map systolic architectures onto stacked optoelectronic smart pixel planes connected through free-space optical interconnections. Within a smart pixel plane we exploit array computing (***intra-plane parallelism***) and between the planes we can profit from pipeline processing (***inter-plane parallelism***).

With exception of the clock distribution the communication in purely systolic architectures takes place only between few and nearest neighboured processing nodes.

This is very beneficial for VLSI circuits, where long lines cause a lot of problems due to difficult wiring and constant RC delay factors. Hence, systolic approaches that try to avoid global interconnections are very attractive for VLSI circuits.

For optoelectronic 3-D circuits a new situation is given. The latency in the inter-plane communication between different optical path lengths is minimal and can therefore be neglected. A moderate global interconnection scheme in the *inter-plane communication* should not be avoided if this yields in a performance increase. For the *intra-plane communication*, i.e. for the design of the optoelectronic smart pixel circuits, the situation is the same as in 2-D VLSI. Hence, we favour the following recommendations for the development of 3-D optoelectronic smart pixel systems:

- For the design of the 2-D optoelectronic smart pixel circuits purely systolic principles should be applied
- Necessary fan-out and global interconnection schemes should be shifted in the 3rd dimension between the stacked 2-D optoelectronic circuits to make use of the capabilities of optical interconnections.

Pursuing these design rules we can improve the performance compared with 2-D VLSI circuitry in terms of throughput and latency.

3 Transformation Techniques for 2-D Circuits

As next we present three methods to map the array of Fig. 2 onto different 3-D systems, which differ in computing performance and necessary hardware requirements.

3.1 Stacking of 2-D Circuits

The easiest way to transform a 2-D circuit into a 3-D system is the duplication of the planar circuit along the vertical direction, as shown for example in Fig. 3.

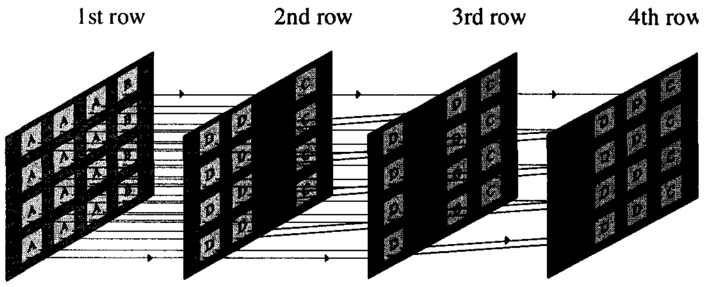


Fig. 3: Vertical stacking of the 2-D circuit of Fig. 2

Within the 2D circuit the planar structure should be divided at certain points in several parts. The electronic connections between these parts are removed by optical ones. For example the separation points could be set up along the rows or the columns of the systolic array. Either the cells of a row or a column are duplicated and integrated on one integrated optoelectronic circuit. In addition this simplifies the design of the resulting optoelectronic circuits because the processing nodes in a row or in a column of a systolic array are frequently identical or show only minor differences. This mapping process is repeated for all rows or columns. The resulting planes are stacked in a 3-D architecture in the same order as it is defined by the order of successive rows or columns in the 2-D array.

If this mapping process is applied to the adder array of Fig. 2 we get as result a 3-D system. If we can double the rows or columns m times, the 3-D system will supply m times more results than the 2-D circuit. Hence, with the 3-D system we can improve the throughput by the factor m. The real value for m depends on technological parameters as the integration density or the number of possible optical pads. Due to the unchanged systolic data flow this kind of mapping does not improve the latency, i.e. the time needed to carry out a single addition is still $3\sqrt{n}-3$ or of order $O(\sqrt{n})$. To improve latency we have to take out a redesign of the 2-D layout structure.

3.2 Mapping the 2-D circuit onto a 3-D system to optimise latency

Another way to improve the performance is the modification of the 2-D VLSI algorithm itself. In contrast to the transformation technique shown before this can not be considered as a generic applicable procedure. The idea is to separate the operand bits of wordlength n not in $\sqrt{n}$ partitions as in the 2-D case, but in $\sqrt[3]{n}$ partitions with $\sqrt[3]{n}\ x\ \sqrt[3]{n}$ bits per partition. Each partition is processed simultaneously on a separate plane. An example for such a partitioning for the case n=8 shows Fig. 4.

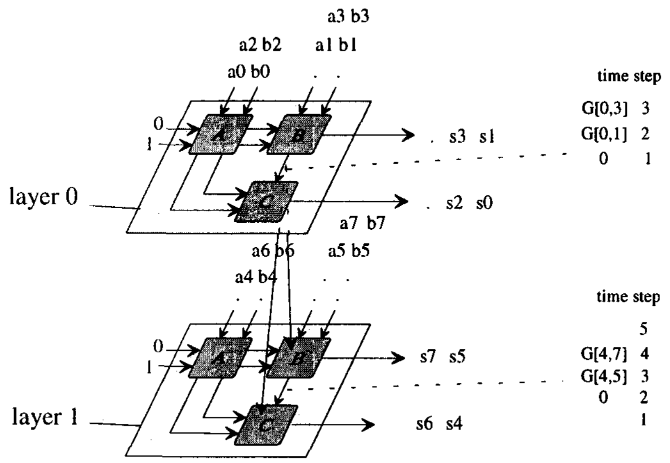


Fig. 4: 3-D reorganisation of the 2-D array to minimise latency

In each layer the same procedure is carried out as it is proposed for the 2-D case. Then the generate bits $G[q\cdot\sqrt[3]{n},(q+1)\cdot\sqrt[3]{n}-1]\,;q\in[0,\sqrt[3]{n}-1]$ are sent to the next layer. To transfer the generate bits between the layers we need a broadcast (s. Fig. 4.). The latency can be determined as follows: In each layer we work on the partitions in the same way as in Kühnel's 2-D array. Hence, after $\sqrt[3]{n}+1$ clock cycles we have calculated the generate bit of the initial partition $G[0,\sqrt[3]{n}-1]$. By exploiting pipeline mechanisms between all planes the generate bits of successive partitions can be calculated step by step.

Decisive for the latency is when the generate block bit $G[0,(\sqrt[3]{n}-1)\cdot\sqrt[3]{n}\cdot\sqrt[3]{n}-1]$ arrives in the last plane. This is performed after $\sqrt[3]{n}+1+\sqrt[3]{n}-1$ steps. Then in additional $\sqrt[3]{n}-1$ steps the sum bits of the last layer are calculated and shifted out of the array. Hence, we need $3\cdot\sqrt[3]{n}-1$ steps; see the example for n=8 in Fig. 4. Hence, in the 3-D case the latency is of $O(\sqrt[3]{n})$ compared with $O(\sqrt{n})$ in the 2-D case.

3.3 Mapping the 2-D Circuit onto a 3-D System to Optimise Period Time

Opposed to the solution described in 3.2 the quadratic structure of the original array is preserved. We receive a 3-D structure in which a quadratic data plane of size $\sqrt{n}x\sqrt{n}$ containing the operand bits is manipulated by moving like a wave through the system. Fig. 5 displays such an architecture for the example of n=16.

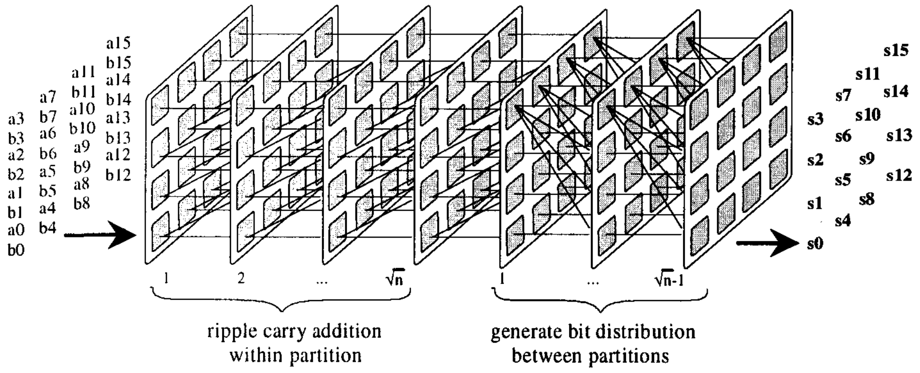

a3 b3 a2 b2 a1 b1 a0 b0
a7 b7 a6 b6 a5 b5 a4 b4
a11 b11 a10 b10 a9 b9 a8 b8
a15 b15 a14 b14 a13 b13 a12 b12
s3 s2 s1 s0
s7 s6 s5 s4
s11 s10 s9 s8
s15 s14 s13 s12
1 2 ... √n
1 ... √n-1
ripple carry addition within partition
generate bit distribution between partitions

Fig. 5: 3-D reorganisation of the 2-D array to minimise period time

In none of the planes intermediate results are stored. Hence, we can shift in each time step a new data plane into the system and reduce the period time to the optimum of 1. In the first $\sqrt{n}$ planes the intermediate sum bits and the necessary block generate bits $G[q\cdot\sqrt{n},(q+1)\cdot\sqrt{n}-1]\,;q\in[0,\sqrt{n}-1]$ are calculated with the ripple-carry-technique. The following $\sqrt{n}-1$ planes serve to distribute one after the other the generate bits from the least significant partition to the most significant one and to determine the final sum bits. Hence, the latency is $2\sqrt{n}-1$ and therefore of the same order $O(\sqrt{n})$ as in the 2-D case.

The gate logic inclusive the exact wiring between successive smart pixel planes is too extensive to represent here and can therefore be found in [9]. The price for optimising the period time are broadcast connections between neighboured planes, which should be realised by means of optics.

4 Comparative Performance Evaluation

As next we compare the determined results for the 3-D systems and the 2-D solution in terms of latency and period time, s. Table 1. Remember that latency and period time for the 2-D solution and the method of vertical stacking are the same. Hence, for a comparison with the 2-D array we can use the figures for the method described in 3.1. Because all nodes in each of the three possibilities (3.1 to 3.3) have the same gate stages to propagate, it is justified to compare only the number of discrete time steps without loss of generality.

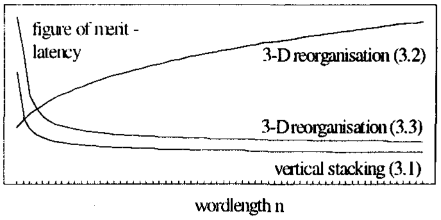
Table 1: Performance comparison of 2-D and 3-D solutions

			period time P_n		# smart pixel planes L_n
vertical stacking (3.1)	$3\sqrt{n}-3$	$O(\sqrt{n})$	$\sqrt{n}+1$	$O(\sqrt{n})$	$\sqrt{n}$
3-D reorganisation to minimise latency (3.2)	$3\sqrt[3]{n}-1$	$O(\sqrt[3]{n})$	$\sqrt[3]{n}+1$	$O(\sqrt[3]{n})$	$\sqrt[3]{n}$
3-D reorganisation to minimise period time (3.3)	$2\sqrt{n}-1$	$O(\sqrt{n})$	1	$O(1)$	$2\sqrt{n}-1$

To evaluate the three 3-D solutions with one another we define figure of merits for the latency ρ_T, and the period time ρ_P (5). The cost for using the 3rd dimension is measured in the number of necessary planes L_n. The fewer layers we need the more favourable are the impacts on the hardware requirements. Hence, we write L_N in the denominator. The same yields for the number of steps for P_n and T_n. The computing power is measured in the wordlength n. Hence, we write the wordlength n in the nominator.

$$\rho_T = \frac{n}{T_n \cdot L_n} \qquad \rho_P = \frac{n}{P_n \cdot L_n} \quad . \tag{5}$$

The course of the curve of ρ_T (s. Fig. 6 left) shows unambiguous the advantage of the 3-D architecture we get with the method presented in 3.2. Only for this architecture the figure of merit increases with the wordlength. The right curve of Fig. 6 shows the superiority of the method presented in 3.3.



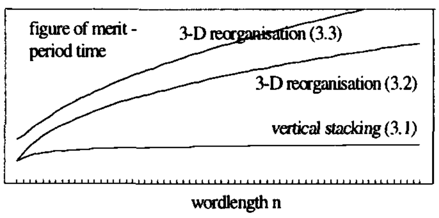


Fig. 6: Course of the figure of merits for latency and period time

In 3-D circuits based on smart pixels we aspire to parallel processors with in space distributed parallel pipeline units. How many operations are possible depends mainly on the number of optical pads that can be realised. For smart pixel circuits based on hybrid SEED technology an interconnection density of about 15000 optical pads per cm^2 with up to 400 MHz cycle time is state-of-the-art. The processing elements of our 3-D solutions need 8 pads each. Hence, for the 3-D circuit described in 3.3 ($P_n=1$) we can get a maximum peak performance of $15000/(8*32)*400MHz \approx 23$ GIPS measured in 32-bit additions. This is much more than current microprocessors offer.

5 Conclusion

In this paper we presented a transformation technique to map 2-D VLSI circuits onto 3-D circuits using optical interconnections. These 3-D circuits are based on optoelectronic smart pixel technology. The mapping was carried out for the example of a purely systolic adder, which was evaluated as optimal for integration in planar VLSI technology. Three different mapping methods have been investigated. It was shown that both latency and period time of the 2-D solution could be improved with either a latency minimising or a period time minimising 3-D smart pixel architecture.

The goal of the future work must be the practical realisation of the proposed concepts for 3-D circuits as well as the extension of the adder towards a scaleable integer processor architecture. Concerning the second point we can base on own earlier carried out work on systolic arrays and digital optical architectures [10], [11].

References

[1] Miller, D.A.B.: Hybrid SEED-Massively Parallel Optical Interconnections for Silicon ICs. Proc. MPPOI'95 San Antonio IEEE CS Press (1995)

[2] Jahns, J., Acklin, B.: Integrated planar optical imaging systems with high interconnection density. *Opt. Letters* 18 1993 1594-1596

[3] Irakliotis, L.J., et al.: Optoelectronic Parallel Processing with Surface-Emitting Lasers and Free-Space Interconnects. *Journal of Lightwave Technology* vol. 13 No. 6 June 1995 1074-1084

[4] Lentine, A.L., et. al.: Evolution of the SEED technology: Bistable logic gates to optoelectronic smart pixels. *IEEE J. Quan. Elect.*,vol.29, 1993 655-669

[5] Kühnel, L.: Optimal purely systolic addition. in: Proc. ARITH-10 Grenoble IEEE CS Press (1991)

[6] Hwang, K.: *Computer Arithmetic - Principles, Architecture and Design.* Wiley New York 1979

[7] Koren, I.: *Computer Arithmetic Algorithms.* Prentice Hall Inc. Englewood Cliffs New Jersey 1993

[8] Omondi, A.R.: *Computer Arithmetic Systems, Algorithms, Architecture and Implementation.* Prentice Hall Inc. Englewood Cliffs New Jersey 1994

[9] Fey, D.: A comparison study between 2-D VLSI circuits and 3-D circuits based on multifunctional smart pixels, accept. for ICAPT'96 Montreal 1996

[10] Fey, D., Brenner, K.H.: Digital Optical Arithmetic based on Systolic Arrays and Symbolic Substitution Logic. *Journ. of Opt.Comp.* vol. 1 1990 153-167

[11] Erhard, W., Fey, D.: *Parallele digitale optische Recheneinheiten.* Teubner Verlag Stuttgart 1994