Resilient data encoding for fault-prone signal transmission in parallelized signed-digit based arithmetic

David Neuhäuser and Eberhard Zehendner

Institute of Computer Science
Friedrich Schiller University
D-07737 Jena, Germany
{david.neuhaeuser,nez}@uni-jena.de

Abstract: When arithmetic components are parallelized, fault-prone interconnections can tamper results significantly. Constantly progressing technology scaling leads to a steady increase of errors caused by faulty transmission. Resilient data encoding schemes can be used to offset these negative effects. Focusing on parallel signed-digit based arithmetic frequently used in high-speed systems, we propose suitable data encodings that reduce error rates by 25%. Data encoding should be driven by the occurrence probabilities of digits. We develop a methodology to obtain these probabilities, show an example fault-tolerant encoding, and discuss its impact on communicating parallel arithmetic circuits in an example error scenario.

1 Introduction

In times of billion-transistor processors being commercially available and transistors being processed in 22 nanometer CMOS process [ITR11], it becomes more and more difficult to design fault tolerant [NSF01, RSKW07] and mixed critical systems [PMN+09]. More complex circuits require increased inter- and intra-circuit connections which become increasingly fault-prone.

Focusing on fast, parallelized, signed-digit based arithmetic, used extensively for instance in CORDIC arithmetic, we propose a data encoding which can significantly lower transmission error rates. Our data encoding principle is based on occurrence probabilities of digits. We show that digit probabilities in signed-digit arithmetic converge when results of addition operations are iteratively reused as input to other addition operations. Digits with the highest limit probability should have more than one bit level encoding. Some errors at bit level would result in unchanged values at digit level. We apply our methodology exemplarily to 2-bit encodings and provide an error rate optimal encoding.

Alternative approaches like using check symbols have been proposed [COP+06], which are less efficient in terms of latency, since every arithmetic operation has to be done multiple times to obtain error information.

In the following section we discuss the signed-digit arithmetic used. In Section 3 we show our methodology to obtain digit probabilities for signed-digit encoded data. In Section 4

we discuss a possible communication error scenario, where fault tolerant data encoding can reduce error probabilities, and give recommendations for error resilient encoding. Applying our methodology, we provide accurate data word probabilities for common signed-digit adder cell implementations in Section 5 and present error rates for different encoding schemes. We conclude in Section 6 and give an outlook to future work.

2 Signed-digit arithmetic

A special case of a signed-digit [Avi61] number system is a signed-binary number system, where each digit is limited to $\{-1, 0, 1\}$. In the following we focus on signed-binary number systems. A signed binary number is defined as

$$Z_{sb} = (z_{n-1}, ..., z_0), z_i \in \{-1, 0, 1\}, 0 \leq i < n \quad (1)$$

$$I(Z_{sb}) = \sum_{i=0}^{n-1} 2^i \cdot z_i \quad (2)$$

where $z_i \in \{-1, 0, 1\}$ and $I : \{-1, 0, 1\}^n \to \mathbb{Z}$ is the interpretation function.

A signed-binary adder (SBA) calculates $S_{sb} = A_{sb} + B_{sb}$ which corresponds to $S = A + B$ with $A, B, S \in \mathbb{Z}$, $I(A_{sb}) = A$, $I(B_{sb}) = B$, and $I(S_{sb}) = S$. We decompose this into digit operations:

$$S_{sb} = A_{sb} + B_{sb} \quad (3)$$

$$\sum_{i=0}^{n-1} s_i = \sum_{i=0}^{n-1} 2^i \cdot a_i + \sum_{i=0}^{n-1} 2^i \cdot b_i; s_i, a_i, b_i \in \{-1, 0, 1\} \quad (4)$$

$$= \sum_{i=0}^{n-1} 2^i \cdot (a_i + b_i) \quad (5)$$

Figure 1 shows this decomposition. One operation at digit i calculates $s_i = a_i + b_i$. Since $a_i, b_i \in \{-1, 0, 1\}$, $a_i + b_i \in \{-2, -1, 0, 1, 2\}$, but $s_i \in \{-1, 0, 1\}$, we need some carry to propagate $\{-2, 2\}$ to the digit at $i + 1$. Focusing on a 3-level design as in Chow and Robertson [CR78], we introduce $c_i \in \{-1, 0\}$ and $d_i \in \{0, 1\}$ as a solution. We include c_i and d_i in Equation 5:

$$S_{sb} = \sum_{i=0}^{n-1} 2^i \cdot (a_i + b_i + c_i + d_i - 2 \cdot c_{i+1} - 2 \cdot d_{i+1}) \quad (6)$$

Here c_0 and d_0 are the carry-ins of the whole adder, set to 0 in normal operation. The carry-outs of the whole adder are c_n and d_n. For any i the signed-binary adder cell (SBAC) calculates:

$$s_i + 2 \cdot c_{i+1} + 2 \cdot d_{i+1} = a_i + b_i + c_i + d_i, \quad (7)$$

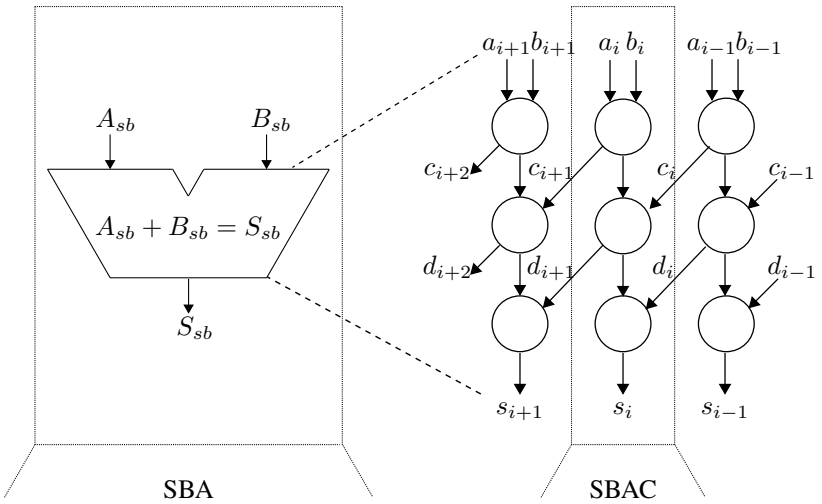

Figure 1: Signed-binary adder (SBA) consisting of three level signed-binary adder cells (SBAC) shown at numerical level, see [CR78, S.111].

The calculation of c_{i+1} must be independent ($\perp$) from c_i and d_i. the calculation of d_{i+1} must be independent from d_i, see again Figure 1. By enforcing these independencies, the remaining carry chain is locally constraint, the calculation of any s_i depends only on $a_i, b_i, a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}$, see also [Zeh92].

3 Digit probabilities

We now describe a SBAC through atomic operations that are in accordance to Equation 8.

$$e_i = a_i + b_i \tag{8}$$

$$c_{i+1}(t, a_i, b_i) = \begin{cases} 0 & \text{when } e_i > 0, \\ -1 & \text{when } e_i < 0, \\ \gamma(t, a_i, b_i) & \text{when } e_i = 0. \end{cases} \tag{9}$$

$$f_i = e_i - 2 \cdot c_{i+1}(t, a_i, b_i) \tag{10}$$

$$g_i = f_i + c_i \tag{11}$$

$$d_{i+1} = \begin{cases} 0 & \text{when } g_i \leq 0, \\ +1 & \text{when } g_i > 0. \end{cases} \tag{12}$$

$$h_i = g_i - 2 \cdot d_{i+1} \tag{13}$$

$$s_i = h_i + d_i = a_i + b_i + c_i + d_i - 2 \cdot c_{i+1} - 2 \cdot d_{i+1} \tag{14}$$

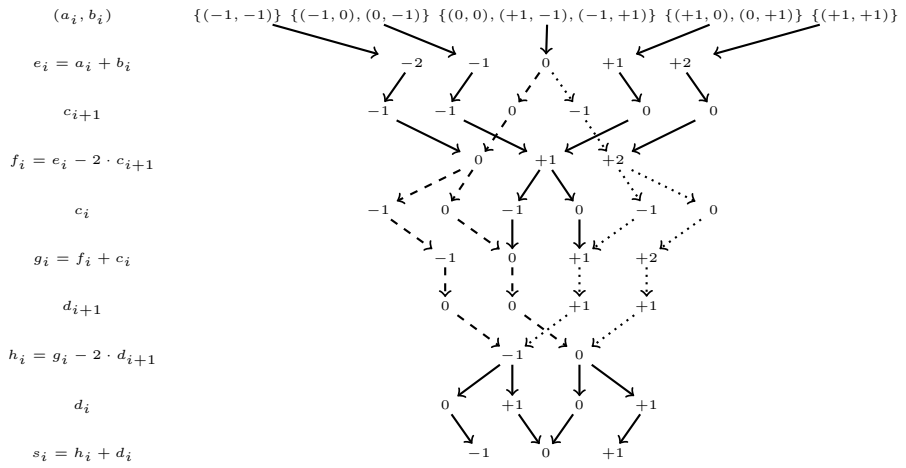

Figure 2: Signed-binary adder cell decision graph. For $a_i + b_i = 0$, the dashed graph denotes a choice of $c_{i+1} = 0$, the dotted graph a choice of $c_{i+1} = -1$. For $a_i + b_i = 0$ it is obvious, that d_{i+1} depends on the choice of c_{i+1} but not on c_i. Furthermore, for $a_i + b_i = 0$,s_i does not depend on the choice of c_{i+1}, but s_{i+1} does depend on the choice of c_{i+1} through d_{i+1}.

We construct a decision graph, see Figure 2, that shows all possible degrees of freedom when constructing a functionally correct SBAC that is constrained by the formal model from Section 2. To enforce the independence constraints , the adder cell has no knowledge of c_i and d_i when calculating c_{i+1}, and no knowledge of d_i when calculating d_{i+1}. Some choices of c_{i+1} and d_{i+1} may therefore be wrong, when worst case values of c_i or d_i occur. In Figure 2, all impossible choices of c_{i+1} and d_{i+1} have already been removed. There are still left some degrees of freedom in choosing c_{i+1} and d_{i+1}, but by fixing a choice on c_{i+1}, we lose all freedom of choice in d_{i+1}. We illustrated the choice of $c_{i+1} = -1$ by dotted arrows and of $c_{i+1} = 0$ by dashed arrows. We see by the dotted and dashed paths, that this choice also fixes the decision of d_{i+1}.

Our SBAC model offers $2^3 = 8$ different signed-binary adder cells at the numerical level. Let t be the *type id* of the design choice, $0 \leq t < 2^3$. All possible design choices are

t	$\gamma(t,0,0)$	$\gamma(t,+1,-1)$	$\gamma(t,-1,+1)$
0	0	0	0
1	0	0	-1
2	0	-1	0
3	0	-1	-1
4	-1	0	0
5	-1	0	-1
6	-1	-1	0
7	-1	-1	-1

Table 1: Meaning of parameter t in description of SBAC.

l_i	z	$P(l_i = z)$
e_i	-2	$P(a_i = -1) \cdot P(b_i = -1)$
	-1	$P(a_i = 0) \cdot P(b_i = -1) + P(a_i = -1) \cdot P(b_i = 0)$
	0	$P(a_i = 0) \cdot P(b_i = 0) + P(a_i = -1) \cdot P(b_i = +1) +$
		$P(a_i = +1) \cdot P(b_i = -1)$
	$+1$	$P(a_i = 0) \cdot P(b_i = +1) + P(a_i = +1) \cdot P(b_i = 0)$
	$+2$	$P(a_i = +1) \cdot P(b_i = +1)$
c_{i+1}	-1	$P(e_i = -2) + P(e_i = -1) + P(\gamma(t, a_i, b_i) = -1)$
	0	$P(e_i = +1) + P(e_i = +2) + P(\gamma(t, a_i, b_i) = 0)$
f_i	0	$P(e_i = -2) + P(\gamma(t, a_i, b_i) = 0)$
	$+1$	$P(e_i = -1) + P(e_i = +1)$
	$+2$	$P(e_i = +2) + P(\gamma(t, a_i, b_i) = -1)$
g_i	-1	$P(f_i = 0) \cdot P(c_i = -1)$
	0	$P(f_i = 0) \cdot P(c_i = 0) + P(f_i = +1) \cdot P(c_i = -1)$
	$+1$	$P(f_i = +2) \cdot P(c_i = -1) + P(f_i = +1) \cdot P(c_i = 0)$
	$+2$	$P(f_i = +2) \cdot P(c_i = 0)$
d_{i+1}	0	$P(g_i = -1) + P(g_i = 0)$
	$+1$	$P(g_i = +1) + P(g_i = +2)$
h_i	-1	$P(g_i = -1) + P(g_i = +1)$
	0	$P(g_i = 0) + P(g_i = +2)$
s_i	-1	$P(h_i = -1) \cdot P(d_i = 0)$
	0	$P(h_i = -1) \cdot P(d_i = +1) + P(h_i = 0) \cdot P(d_i = 0)$
	$+1$	$P(h_i = 0) \cdot P(d_i = +1)$

Table 2: Probability level description of $SBAC_t$.

shown in Table 1. Note that the formula for calculating c_{i+1} depends on the input digits (a_i, b_i) and the chosen design parameter t. Let $SBAC_t$ be the design using choice t to calculate c_{i+1}.

Assigning probability information to the symbols in Equations 8 through 12 we are able to calculate the digit probabilities. At probability level $SBAC_t$ is described as shown in Table 2.

$P(\gamma(t, a_i, b_i) = 0)$ and $P(\gamma(t, a_i, b_i) = -1)$ are calculated in accordance to Table 1 as

$$\begin{aligned} P(\gamma(t, a_i, b_i) = 0) \quad = \quad & P(a_i = 0) \cdot P(b_i = 0) \cdot P(t \in \{0, 1, 2, 3\}) + \\ & P(a_i = +1) \cdot P(b_i = -1) \cdot P(t \in \{0, 1, 4, 5\}) + \\ & P(a_i = -1) \cdot P(b_i = +1) \cdot P(t \in \{0, 2, 4, 6\}) \end{aligned}$$

$$
\begin{aligned}
P(\gamma(t,a_i,b_i)=-1) &= P(a_i=\ \ 0)\cdot P(b_i=\ \ 0)\cdot P(t\in\{4,5,6,7\}) + \\
&\quad P(a_i=+1)\cdot P(b_i=-1)\cdot P(t\in\{2,3,6,7\}) + \\
&\quad P(a_i=-1)\cdot P(b_i=+1)\cdot P(t\in\{1,3,5,6\})
\end{aligned}
$$

Note that $P(e_i = 0) = P(\gamma(t, a_i, b_i) = 0) + P(\gamma(t, a_i, b_i) = -1)$.

4 Digit error scenario

Figure 3 shows two circuits exchanging digits by signal lines 0 through $n-1$. On each line, the signal is sent as $l_i \in \{0,1\}$ and received as $T(l_i) \in \{0,1\}$. The digits received may differ from the digits sent due to imperfect wiring [SOHH07, KPKJ07].

In our simple error model, p_{bf} denotes the probability, that one bit is inverted. The possibility of a bit flip leads to

$$
\begin{aligned}
T(l_i) &= \begin{cases} l_i & \text{when no bit flip occurred,} \\ 1-l_i & \text{else.} \end{cases} \\
P(T(l_i)=l_i) &= 1-p_{bf}
\end{aligned}
$$

When encoding signed-binary digits $\{-1, 0, 1\}$ with two bits, we can leave one bit combination unused or encode one of the digit values by two different bit combinations. We call the first non-redundant, the second redundant encoding. When using redundant encoding, our SBAC outputs only one code for the double encoded digit. The other code can only occur by faulty transmission, but is interpreted as a correct double encoded digit. Table 3 shows the effects of using a 2-bit redundant encoding of signed-binary digits with such a correction in comparison to a non-redundant encoding with no error correction. $T_{nr}(dw)$ is the result of transporting the 2-bit data word dw with non-redundant encoding and no error correction, $T_r(dw)$ is the result of transporting the data word with redundancy and error correction.

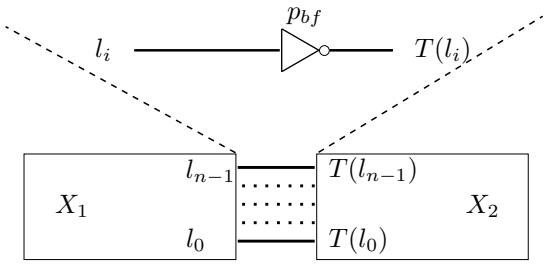

Figure 3: Circuit X_1 communicates with circuit X_2 through signal lines l_0 to l_{n-1}. Our simple error model consists of a possible bit flip with probability p_{bf}.

	no redundancy		redundancy	
example encoding	data word	error prob.	data word	error prob.
00	dw0	$2 \cdot p_{bf} - p_{bf}^2$	dw0	p_{bf}
01	unused	—	defected dw0	—
10	dw1	$2 \cdot p_{bf} - p_{bf}^2$	dw1	$2 \cdot p_{bf} - p_{bf}^2$
11	dw2	$2 \cdot p_{bf} - p_{bf}^2$	dw2	$2 \cdot p_{bf} - p_{bf}^2$

Table 3: Simple error model: Error reduction by using gray code adjoined encoding for data words $dw0$, $dw1$, and $dw2$. All calculations using the redundant encoding consist of a 01 to 00 correction.

The probability of an uncorrected error in $dw0$ in our example is reduced

$$P(T_{nr}(dw0) \neq dw0) = 2 \cdot p_{bf} - p_{bf}^2 \quad > \quad 2 \cdot p_{bf} - p_{bf} = p_{bf} \\ = P_{red}(T_r(dw0) \neq dw0) \tag{15}$$

The error probability for any 2-bit data word $dw \in \{dw0, dw1, dw2\}$ with no error correction can be calculated as

$$\begin{aligned} P(T_{nr}(dw) \neq dw) &= P(dw = dw0) \cdot (2 \cdot p_{bf} - p_{bf}^2) + P(dw = dw1) \cdot \\ &\quad (2 \cdot p_{bf} - p_{bf}^2) + P(dw = dw2) \cdot (2 \cdot p_{bf} - p_{bf}^2) \\ &= (P(dw = dw0) + P(dw = dw1) + P(dw = dw2)) \cdot \\ &\quad (2 \cdot p_{bf} - p_{bf}^2) \\ &= 1 \cdot (2 \cdot p_{bf} - p_{bf}^2) = 2 \cdot p_{bf} - p_{bf}^2 \end{aligned}$$

In comparison to an applied error correction

$$\begin{aligned} P(T_r(dw) \neq dw) &= P(dw = dw0) \cdot p_{bf} + P(dw = dw1) \cdot \\ &\quad (2 \cdot p_{bf} - p_{bf}^2) + P(dw = dw2) \cdot (2 \cdot p_{bf} - p_{bf}^2) \\ &= P(dw = dw0) \cdot p_{bf} + (P(dw = dw1) + P(dw = dw2)) \cdot \\ &\quad (2 \cdot p_{bf} - p_{bf}^2) \end{aligned}$$

With equation 16 we get

$$P(T_{nr}(dw) \neq dw) \quad > \quad P(T_r(dw) \neq dw)$$

The encoding strategy is rather simple: Use the redundant encoding $00, 01$ to encode the digit with the highest probability of occurrence to reduce the error probability. The error ratio of this strategy can be calculated by

$$\begin{aligned} \text{error ratio (of red dw encoding)} &= \frac{\text{error rate of dw red encoding}}{\text{error rate of dw non-red encoding}} \\ e(dw) &= \frac{P(T_r(dw) \neq dw)}{P(T_{nr}(dw) \neq dw)} \end{aligned} \tag{16}$$

5 Results

For any trivial digit probability, where one symbol out of $\{-1, 0, 1\}$ has a probability of 1, and the others have of 0, the probabilities of the output symbols are either 0.0 or 1.0. If any other, non-trivial digit probability is applied to initial a_i,b_i, and the probability distribution of s_i is looped back to the $SBAC_t$ inputs a_i,b_i, the probabilities converge, see for example Figure 4, where $t = 5$ and initially $P(a_i = 0) = P(b_i = 0) = 0.1$ and $P(a_i = 1) = P(b_i = 1) = 0.9$.

Since the calculation of c_{i+1} (and indirectly d_{i+1}) depends on t, the converg also depends on t, see Figure 5.

A sample application for signed-digit arithmetic could be a CORDIC-based algorithm. CORDIC [Vol59] transforms initial data iteratively with predefined coefficients. Let A_0 be the initial N-bit data, and B_i the predefined coefficients:

$$A_{i+1} = f_{cordic}(A_i, B_i) \text{ , with } 0 \leq i \leq N - 1 \tag{17}$$

f_{cordic} is the CORDIC function for processing A_i and B_i by an adder/subtracter and shifter. A_N is the final result, A_0 the input data to be processed. To simulate the impact of subtraction, we assume a probability of 50%, that input for b_i has opposite signs. Figure 6 corresponds to this more realistic use case.

Applying the simple error model to a SBAC type 7 with a probability of 50% of $+/-$ alternation, we investigate the digit error depending on the bit flip error and the encoding, as shown in Figure 7. The error ratio is 75% for small p_{bf} and increases to expected 100% for $p_{bf} = 1$. This means by using error correction and redundant encoding for digit 0

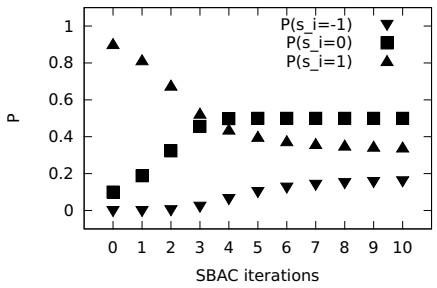

Figure 4: SBAC adder operation data probability for $t = 5$ and initial $P(a_i = 0) = P(b_i = 0) = 0.1$ and $P(a_i = 1) = P(b_i = 1) = 0.9$.

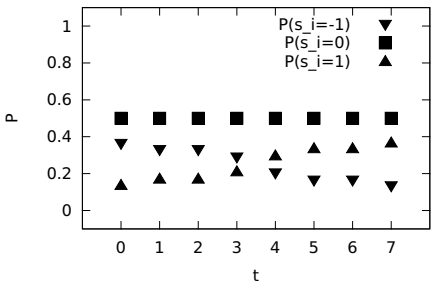

Figure 5: SBAC adder operation data probability depending on t for non-trivial initial data probabilities.

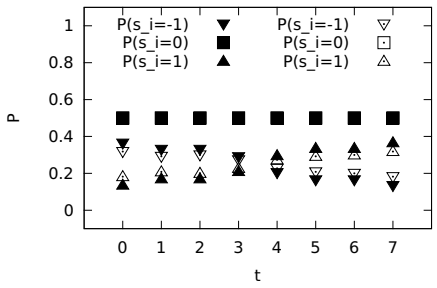

Figure 6: SBAC adder/subtracter data probability depending on type t for non-trivial initial data probabilities. Solid symbols represent Figure 5, hollow symbols represent a probability of 50% for $+/-$ alternation.

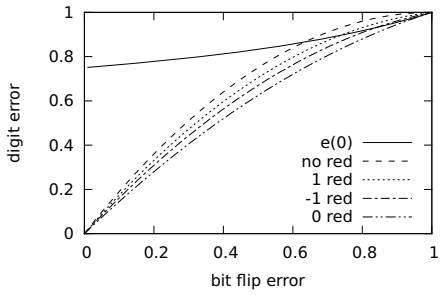

Figure 7: SBAC adder/subtracter digit error probability for $t = 7$ and non-trivial initial digit occurrence. Probability of $+/-$ alternation is 50%. "no red" denotes no redundant encoding, "-1 (0,1) red" denotes redundant encoding and error correction for -1 (0,1), "$e(0)$" denotes the error ratio of redundant encoding and error correction of digit "0", see equation 16.

6 Conclusion and future work

In a scenario, where data paths between memory and a SBAC as well as between several SBACs are fault-prone, the knowledge of digit probabilities offers a chance to use a data encoding scheme that provides some implicit fault tolerance. We have shown a model to gain data flow probability information for signed-binary based arithmetic operations and have proposed a data encoding scheme that provides advanced fault tolerance properties.

In our example, the proposed $SBAC_t$ tends to generate symmetric $P(s_i = -1)$ and $P(s_i = +1)$ probabilities with respect to type t, see again Figure 5. The actual values of $P(s_i = -1)$ and $P(s_i = +1)$ are due to the choice of $c_{i+1} \in \{-1, 0\}$ and subsequently $d_{i+1} \in \{0, +1\}$. Let us call such a design $SBAC_{-t}$.

In contrary, $SBAC_{+t}$ with $c_{i+1} \in \{0, +1\}$ and $d_{i+1} \in \{-1, 0\}$ produces the opposite probability behavior for $P(s_i = -1)$ and $P(s_i = +1)$ with respect to t. Since the probabilities $P(s_i = -1)$ and $P(s_i = +1)$ are symmetric, $SBAC_{-,t_1}$ and $SBAC_{+,t_2}$ with $t_1 + t_2 = 7$ have the same s_i digit probabilities. The digital circuit designer is free to chose the more implementation friendly design.

Still, more detailed research is needed. Changing the design from using one $SBAC_-$ (or a chain of n $SBAC_-$) to the use of alternating $SBAC_-$ and $SBAC_+$ will lead to digits with $P(s_i = -1) = P(s_i = 1)$. The possible advantages for fault tolerance and reduced (false) carry generation c_n and d_n have to be investigated.

The mentioned application, i.e. CORDIC, is not very accurately described, since the coefficients B_i are actually calculated and saved to some memory in advance. An arbitrary encoding can be chosen here to enforce a desired data probability characteristic, making the whole system even more fault-tolerant, especially when allowing more than two bits for encoding one digit.

References

[Avi61] Algirdas A. Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, 10(3):389–400, Sep 1961.

[COP+06] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano. Localization of Faults in Radix-n Signed Digit Adders. In *Proceedings of the 12th IEEE International On-Line Testing Symposium*, IOLTS, pages 178–180, Washington, DC, USA, 10–12 Jul 2006. IEEE Computer Society.

[CR78] C. Y. Chow and J. E. Robertson. Logical Design of a Redundant Binary Adder. *Proc. 4th Symposium on Computer Arithmetic*, pages 109–115, 1978.

[ITR11] ITRS. *International Technology Roadmap for Semiconductors. 2011 Edition. Emerging Research Devices*. http://www.itrs.net/links/2011itrs/2011Chapters/2011ERD.doc, 2011.

[KPKJ07] Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K. Jha. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 150–161, New York, NY, USA, 2007. ACM.

[NSF01] K. Nikolic, A. Sadek, and M. Forshaw. Architectures for Reliable Computing with Unreliable Nanodevices. In *Proc. 1st IEEE Conference on Nanotechnology*, pages 254–259, 2001.

[PMN+09] Rodolfo Pellizzoni, Patrick Meredith, Min-Young Nam, Mu Sun, Marco Caccamo, and Lui Sha. Handling mixed-criticality in SoC-based real-time embedded systems. In *Proceedings of the seventh ACM international conference on Embedded software*, EMSOFT '09, pages 235–244, New York, NY, USA, 2009. ACM.

[RSKW07] Warren Robinett, Gregory S. Snider, Philip J. Kuekes, and R. Stanley Williams. Computing with a trillion crummy components. *Commun. ACM*, 50(9):35–39, Sep 2007.

[SOHH07] M. Stahl-Offergeld, H.-P. Hohe, and M. Hackner. Spinning current offset in vertical Hall sensors caused by imperfect wiring. In *Proc. 13th International Sensor Conference*, volume 2 of *SENSOR 2007*, pages 211–216, 22–24 May 2007.

[Vol59] Jack E Volder. The CORDIC Trigonometric Computing Technique. *Ieee Transactions On Electronic Computers*, EC-8(3):330–334, 1959.

[Zeh92] Eberhard Zehendner. Reguläre parallele Addierer für redundante binäre Zahlsysteme. Technical report, Report 255, Institut für Mathematik der Universität Augsburg, Juni 1992.