A Parameterizable Fault Simulator for Bridging Faults

Piet Engelke Bernd Becker Martin Keim*

Institute of Computer Science
Albert-Ludwigs-University
Am Flughafen 17
79110 Freiburg im Breisgau, Germany
<engelke, becker>@informatik.uni-freiburg.de

Infineon Technologies AG
IT COM TI TPE
Balanstrasse 73
81541 München
martin.keim@infineon.com

Abstract

We present the concept of a multiple-valued logic simulator that is able to more accurately determine the possible behavior of a circuit in the presence of a bridging faults. By a user defined mapping of a range of voltages to a logic value the simulator takes care of certain voltages more closely than common bridge fault simulators that map all voltages to either logic 1 or 0. Experimental results are given to demonstrate the improved fault detection possibilities.

1 Introduction

For current CMOS designs bridges, i.e. an unintentional connection between lines that are normally not connected, are a common physical defect. Usually, only bridges between the outputs of gates are considered. The impact of such a connection on the behavior of the circuit depends on its resistance. Bridges with high resistance (resistive short) cause a delayed transition at the output of the succeeding gates resulting in a delay fault. Bridges with low resistance (hard short) change the static behavior of the circuit. For the later one many models have been invented for the gate level: The wired and the dominant models [1], although good models in TTL, are no longer valid in CMOS [11, 4]. Also a composition of stuck-at faults has been considered to model the behavior of a bridge, e.g. [17]. However, the most common model is the biased voting model [14] that is an improvement of the voting model [16]. These models view the bridge as a resistive divider between V_{DD} and GND when the gates try to drive the shorted lines to different values. The resulting voltage is then mapped to either the logic value 1 or 0. In biased voting the mapping is done with respect to the different thresholds of the succeeding gate inputs. This approach handles the faulty part of the circuit as a linear device, although transistors are not linear devices. However, the resistor network can accurately model the relative strength of the transistors involved.

For simulating the circuit different methods have been developed to accurately determine the voltage at the bridge. There are for example table lookup based simulation methods that determine the correct output voltage for every pair of gate types in the library and every possible input combination [18, 8] in a preprocessing phase e.g. using extensive SPICE simulations. Another method is a mixed-mode simulation that for example determines the bridge voltage with a simulation at electrical or transistor level at the fault location and its vicinity [12]. These methods accurately determine the bridge voltage, but have three problems besides the memory demand and the execution time of the algorithms:

(i) According to the fault model, the voltage has to be mapped to a logic value. This annihilates the information gained from the highly accurate voltage computation since e.g. all voltages that are (clearly) above the HIGH threshold are mapped to logic 1. However, the output voltage of a gate having one or more input voltages shortly above the threshold might not correspond clearly to a logic value. Nevertheless, this effect might be blocked after a number of gates.

(ii) Due to variations in the manufacturing process the individual instances of the circuit are not identical (see e.g. [4]). This leads e.g. to the problem what to do, when the computation results in a voltage that is 1/100 V beneath the theoretical threshold of the gate? Some gates might not switch, others might do so. However, the output value will be considerably delayed. At this level of accuracy, the relation between the model and the real circuit is no longer given.

From (i) and (ii) it follows, that there is no need for a highly accurate computation of the bridge voltage. This is also underlined by the results of [19]. However, a good approximation that takes manufacturing variances into account would be more suitable.

(iii) As feature size is scaled down, the resistance per unit length goes up almost quadratically. Higher bridge resistance combined with lower device resistance during ON state result in an increasing number of resistive faults (delay faults) [20]. Therefore, more intermediate voltages are discovered at the required arrival time of a signal. Not all of them will fall or rise eventually beyond the threshold (causing a delay fault). Therefore, current bridge fault simulators that calculate the bridge voltage according to a resistive divider and therefore do not directly target delay faults, should at least be able to notice a *potential* delay fault and should not ignore the information already computed.

Logic Value	Voltage Range		
4	V_{DD}	–	V_4
3	V_4	–	V_3
2	V_3	–	V_2
1	V_2	–	V_1
0	V_1	–	V_{GND}

Table 1. Voltage-to-logic mapping for the bridge values of Table 2.

There is only a small number of simulators that can handle such problems. For example in [15] a three-valued logic $\{0, X, 1\}$ is used with 0 denoting voltages below the LOW threshold and 1 denoting voltages above the HIGH threshold. All voltages in between are mapped to

*This work has been done while Martin Keim was with the chair of Prof. Becker.

p transistor		n transistor				
		single transistor	parallel network		serial network	
		1	2	3	2	3
single transistor	1	2	1	0	3	4
parallel network	2	3	2	1	4	4
	3	4	3	2	4	4
serial network	2	1	0	0	2	3
	3	0	0	0	1	2

p transistor		n transistor				
		single transistor	parallel network		serial network	
		1	2	3	2	3
single transistor	1	1	0	0	2	4
parallel network	2	2	1	1	3	3
	3	2	2	1	3	4
serial network	2	0	0	0	1	2
	3	0	0	0	1	1

Table 2. **Left: Bridge value when $R_p == R_n$. Right: When $R_p == 2R_n$.**

NOR	0	1	2	3	4
0	4	4	1	0	0
1	4	3	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

NOR	0	1	2	3	4
0	4	4	2	0	0
1	4	3	2	0	0
2	2	2	2	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Table 3. **Left: Normal NOR-operation. Right: NOR-operation throughputting a potential delay.**

X. Similar, but more enhanced, in [13] five logic values are used: $\{0, INT, 1, OSC, X\}$, where INT is used to denote a voltage that is between the LOW and HIGH threshold. (OSC denotes the oscillating line values, and X denotes the unknown value due to incomplete initialization.) However, by mapping all voltages that are not a clear logic 0 or 1 to only one intermediate logic value many information is lost. Furthermore, (ii) and (iii) are not considered by such approaches.

Other interesting approximations are presented in [3] mainly for MOS circuits, [10] that uses a probabilistic fault model, and [9] that is in some way similar to the proposed method.

In the next section the proposed method is introduced. It is pointed out how the it differs from the work of other researchers. Its advantages and limitations are discussed. By a large set of experiments the advantages are underlined in the results section. In the last section some conclusions are drawn and we give some hints for future work.

2 The proposed method

To solve problem (i) and (ii) and to make a first step to attack problem (iii) we propose a bridge fault simulator that performs a simulation with respect to a user-defined mapping of a *(small) voltage range* to a *logic value* of an n-valued logic. The simulation is then performed over the n-valued logic as an approximation of an electrical simulation of the corresponding voltages. Table 1 shows such a mapping. In Table 1 five different voltage ranges are defined. To each of them a logic value is assigned: logic 0 denotes a voltage range that is clearly LOW, logic 1 stands for voltages that are close to the LOW threshold, logic 3 is for voltages around the HIGH threshold, logic 4 denotes a clear HIGH, and finally logic 2 cover all voltages in between the voltages corresponding to logic 1 and 3. In the following, we refer to the lowest (highest) logic value of an n-valued logic as the MIN (MAX) value.

With this approach problem (i) is solved and the occurrence of problem (ii) is considerably reduced. Moreover, the granularity of the approximation is defined by the used logic and the mapping is defined by the user's demands. The mentioned $\{0, X, 1\}$ logic, respectively the $\{0, INT, 1, OSC, X\}$ logic, would correspond to a three-valued logic $\{0, 1, 2\}$ with logic 1 from $\{0, 1, 2\}$ covering all voltages in between the thresholds (OSC and X are not considered in our current implementation). How accurate (iii) is modeled depends on the defined logic operators.

At the fault location the bridge voltage is determined by a table lookup as described later in this section. Since we have a more distinct correspondence between a voltage and a logic value, the "Byzantine General" [2] problem does not occur: How the logic value computed at the bridge site is interpreted by the succeeding gates is left to their evaluation routine and is not the matter of the mapping routine at the bridge site. Therefore, the voting model is sufficient, simplifying the simulation algorithm.

The simulators of [18, 8] built up voltage tables for all gates types in the library and all input combinations. This results in very large tables and several ten hours of SPICE simulation for filling the tables. We consider here a much simpler approach. Since we consider only inter-gate bridges, we first map all gate types to according NAND, NOR, and INV gates [7]. We consider only NAND-2, NAND-3, and NOR-2, NOR-3 and INV as basic gates. Thus, we have to deal only with a very limited number of different parallel and serial networks of p and n transistors.

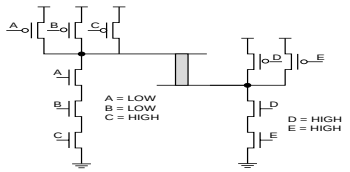

Figure 1. **Two NAND gates connected by a bridge.**

Then, we regard only the relative strength of the pull-up and the pull-down network with respect to the current input combination (similar to [14] and [16]). This means to determine the output logic value of the bridge, we consider only resistances of the connection from V_{DD} via some transistors of the pull-up network, that are connected in parallel or serial, to GND via some transistors of the pull-down network, that are also connected in parallel or serial. In Figure 1 there is an example for two NAND gates having a bridge at their outputs. Note, that also for more complex gates the principles of considereing only the relative strength of the transistor networks driving the output of the gate can be applied. Thus, the presented approach is not limited to only NAND and NOR gates.

For the voltage-to-logic mapping shown in Table 1, Table 2 gives the logic values at the bridge for a 5-valued logic. For simplicity of the examples, the logic values of Table 2 have been determined under the assumption that (i) there is only one type of p and one type of n transistor, and (ii) the p and the n transistors are equally strong (on the left) or, respectively, that a p transistor is modeled by having twice the resistance than a n transistor (on the right). Any other assumption can easily be incorporated

Circuit	Simulation No. faults	Detections	Deteced	Time [s]	Logic Distribution 0	1	2
c0017	29	616	16	0.14	2081	960	2759
c0095	283	10793	205	2.66	100617	17501	79982
c0432	9132	84350	6064	134.27	2237064	291232	3864104
c0499	10000	22571	2208	380.21	15864926	647626	15487448
c0880	10000	88263	3443	219.65	12800144	474728	12725128
c1355	10000	17277	2919	546.74	15871118	610525	15518357
c1908	10000	38696	1977	556.23	12589730	516411	11893859
c2670	10000	81235	2616	563.82	68508353	486295	71005352
c3540	10000	40536	2870	532.92	11862786	720007	9417207
c5315	10000	38089	1620	580.36	67487135	499678	55013187
c6288	10000	396764	9709	1737.28	16491536	1649394	13859070
c7552	10000	36712	1785	620.52	47563368	581260	59855372
cs00027	66	1817	45	0.39	13802	2052	10546
cs00208	6377	44191	1730	52.75	2780248	244856	2714196
cs00298	8348	84086	4308	103.50	10843995	387476	5464529
cs00344	10000	93967	2948	184.65	12826525	534534	12638941
cs00349	10000	94328	2920	172.09	12828448	530725	12640827
cs00382	10000	125683	5875	138.02	18247293	462660	8290047
cs00386	10000	11970	790	96.19	12229016	259964	511020
cs00400	10000	120802	5601	137.47	18250492	459668	8289840
cs00420	10000	63586	2364	98.55	8543641	333342	8123017
cs00444	10000	109405	5270	160.89	18044710	453121	8502169
cs00510	10000	88029	3988	114.14	7381023	402270	5216707
cs00526	10000	102013	5459	132.72	18196073	409266	8394661
cs00641	10000	110969	2155	281.07	22217931	808432	19973637
cs00713	10000	100490	2063	286.05	22024314	775234	19200452
cs00820	10000	32104	4076	130.37	21935650	159215	1905135
cs00832	10000	32015	4201	140.13	21935186	159451	1905363
cs00838	10000	63975	2108	160.88	16904179	329772	15766049
cs00953	10000	120798	3525	220.58	37239681	575673	14184646
cs01196	10000	32369	2099	173.30	20049188	273245	11677567
cs01238	10000	32021	2086	168.33	20055330	271454	11673216
cs01423	10000	69153	2737	281.97	56051352	519125	22429523
cs01488	10000	4695	343	158.98	19743716	243651	5012633
cs01494	10000	4539	361	161.72	19742528	242611	5014861
cs05378	10000	96085	3127	748.35	137946273	721673	89332054
cs1196	10000	6516	315	225.17	20069319	252954	11677727
cs1269	10000	23455	788	528.01	20883070	599371	25517559
cs1512	7917	17083	668	282.57	43774205	300682	17677713
cs3330	10000	68143	1239	610.92	124931424	745917	79322659
cs3384	10000	72516	1158	801.81	117518877	1052831	90428292
cs344	10000	36306	1078	251.85	12767579	618714	12613707
cs4863	10000	57633	1158	809.82	70053944	1009983	48936073
cs6669	10000	107970	1736	1145.06	150737996	1259291	142002713
cs938	10000	62654	2093	143.10	16909591	325243	15765166
Sum	412152	2947268	115844	14976.20	1405055457	22220073	1011535070

Table 4. Results for normal three-valued logic.

into this table. Note, that the logic values (respectively its corresponding voltages) have been verified *afterwards* by a SPICE simulation, with $V_{DD} = 5V$, $V_4 = 3.8V$, $V_3 = 3.2V$, $V_2 = 1.8V$, $V_1 = 1.2V$, and $V_{GND} = 0V$.

To determine the bridge logic value the following steps are performed: Depending on the input vectors and the types of the driving gates the number of the transistors, their type and the connection topology (parallel or serial) are determined. Then, the logic value at the bridge can be found in the according row and column of Table 2. In doing so, the slightly different input pin thresholds of a gate are assumed to lie in the same voltage range. For the bridge fault in Figure 1 Table 2 (left) determines the logic value 4 as follows: In the left NAND gate two of the three p transistors connected in parallel are activated, thus row 6 is selected (p transistor, parallel network, 2 active). The right NAND gate has two serial n transistors activated, thus column 6 is selected (n transistor, serial network, 2 active). When considering a different resistive divider setting (on the right in that table) the bridge's logic value would be 3.

Tables 1 and 2 defined the voltage to logic mapping at the bridge side. Furthermore, for regular gate operation, there are also several possibilities how to define the logic operators, depending on the point of interest. Exemplarily for a NOR-2 operation, in Table 3 (left) there is the regular five-valued logic shown. These logic values have been verified by a SPICE simulation, too. In the rightmost table the logic operation is modified to express a delay fault caused by a delayed input value. The delayed transition originates at the gates succeeding the bridge due to an intermediate voltage at the expected arrival time of the signal. The logic guarantees that the delay is put through the gates - and therefore through the whole circuit - unless a dominating input value terminates the propagation. At a primary output the intermediate logic value (in a five-valued logic this is the logic value 2) denotes that (potentially) a delayed transition is visible.

In the experimental results section, for both variants of the five-valued logic results are shown. Beside these, other definitions of a logic operation might be possible.

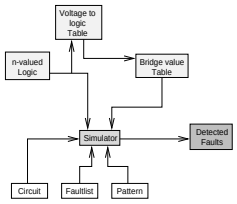

Figure 2. Components of the simulator.

Figure 2 depicts the general architecture of the simulator. The simulation algorithm takes the circuit, a set of input patters, a n-valued logic defining the operators, and a table defining the bridge output logic value. The simulation algorithm itself is not changed when selecting a different logic, modified logic operators or different

	Simulation					Logic Distribution				
Circuit	No. faults	Detections	Prob. Det.	Detected	Time [s]	0	1	2	3	4
c0017	29	900	182	28	0.12	2546	108	0	129	3017
c0095	283	14922	4981	271	2.70	106032	6441	239	2704	82684
c0432	9132	171837	18805	8898	115.86	2322324	37916	16356	8561	4007243
c0499	10000	22571	57337	2208	368.35	15864926	80545	446398	120683	15487448
c0880	10000	265292	24044	9245	192.79	12892132	39799	118246	22955	12926868
c1355	10000	129532	28422	9070	387.44	16163344	44	26959	34050	15775603
c1908	10000	303151	16382	9252	400.92	12862211	2033	7143	19328	12109285
c2670	10000	212177	30924	8001	529.59	68628647	10298	203076	43392	71114587
c3540	10000	375745	19254	8961	400.94	12194619	12993	60857	21516	9710015
c5315	10000	159623	19535	8335	506.98	67601280	11971	239714	27427	55119608
c6288	10000	860647	6257	9959	890.45	17268315	9216	637	204	14721628
c7552	10000	280466	20130	9483	502.41	47808429	9301	89241	29498	60063531
cs00027	66	2131	467	59	0.40	13973	907	294	109	11117
cs00208	6377	94100	66860	4014	54.89	2826925	40384	68735	52579	2750677
cs00298	8348	161391	71078	6780	96.82	10917261	159683	97113	4793	5517150
cs00344	10000	233715	95111	8146	162.47	12980457	95514	94341	87751	12741937
cs00349	10000	234865	92059	8173	161.35	12985851	94685	95398	83891	12740175
cs00382	10000	216749	89637	8663	142.03	18337498	138681	117122	33052	8373647
cs00386	10000	11970	0	790	99.08	12229016	0	259964	0	511020
cs00400	10000	212635	87158	8792	132.72	18345081	132634	119294	34221	8368770
cs00420	10000	123352	93614	4982	95.96	8595595	60695	103331	70841	8169538
cs00444	10000	205385	89104	8926	139.68	18132036	121317	114448	31388	8600811
cs00510	10000	273376	45316	9317	112.21	7523669	26633	12975	51133	5385590
cs00526	10000	182007	94431	8006	134.78	18271240	144483	75937	48600	8459740
cs00641	10000	111931	51243	2189	294.62	22219267	20025	732041	53637	19975030
cs00713	10000	130652	224481	4195	287.48	22039379	49069	348999	320361	19242192
cs00820	10000	77146	19276	7697	119.78	21957478	37059	36080	16972	1952411
cs00832	10000	79141	19589	7652	127.46	21957333	36921	33167	16872	1955707
cs00838	10000	122987	97483	4508	148.14	16942947	64240	102634	73156	15817023
cs00953	10000	385786	63984	7995	207.53	37456755	15941	29058	63238	14435008
cs01196	10000	70594	79941	5026	187.42	20102192	67386	75086	47966	11707370
cs01238	10000	71722	79413	5117	164.23	20108751	67063	71751	48112	11704323
cs01423	10000	119880	79494	6244	274.38	56083580	99303	288558	40559	22488000
cs01488	10000	4695	0	343	156.91	19743716	0	243651	0	5012633
cs01494	10000	4539	0	361	152.83	19742528	0	242611	0	5014861
cs05378	10000	261659	88054	7336	726.22	138125876	128967	211703	49795	89483659
cs1196	10000	6516	0	315	229.36	20069319	0	252954	0	11677727
cs1269	10000	23455	0	788	497.43	20883070	0	599371	0	25517559
cs1512	7917	17083	0	668	270.14	43774205	0	300682	0	17677713
cs3330	10000	69105	21507	1404	636.84	124935270	11754	708963	20522	79323491
cs3384	10000	72516	0	1158	724.81	117518877	0	1052831	0	90428292
cs344	10000	36306	0	1078	273.56	12767579	0	618714	0	12613707
cs4863	10000	57633	0	1158	849.79	70053944	0	1009983	0	48936073
cs6669	10000	107970	0	1736	1159.21	150737996	0	1259291	0	142002713
cs938	10000	122218	92298	4396	146.95	16949086	61078	106718	69893	15813225
Sum	412152	6702073	1987851	231723	13266.03	1409042555	1895087	10692664	1649888	1015530406

Table 5. Results for normal five-valued logic.

bridge-output logic value tables. The simulation is performed at gate level and on n-valued logic values only. Neither sophisticated routines are necessary to determine the bridge voltage, nor a table lookup of this voltage and a succeeding mapping under consideration of the biased voting model for all gate types and input combinations is necessary. This speeds up the simulation process and results in a simulation algorithm that is very close to common stuck-at fault simulation algorithms.

3 Experimental results

To show the advantage of using a more sensitive mapping of voltages to logic values, a number of experiments have been performed. The simulator shown in Figure 2 has been implemented in C++ and has been applied to several ISCAS85 benchmark circuits, the combinational parts of the ISCAS89 benchmark circuits and their addendum [5, 6]. Even a small number of random patterns are sufficient to show the advantage: Only 100 random patterns have been fault simulated on a SUN Ultra-1 workstation.

Due to the logic definition, it is clear that whenever the three-valued logic determines a MIN or MAX value the five-valued logic will do so also. The question is, however, whether *more* MIN or MAX values are computed, or not. Furthermore, if this actually takes place, how does the fault detection change.

To achieve comparable results between the different logics, the fault simulation has to be modified as follows: fault dropping is disabled, and the event driven fault simulation is performed as long as there is an event; it is not stopped due to a detection of a fault at a primary output, since we want to know *all* output changes.

Table 4 to Table 7 show the results. After the circuit name there is the number of considered non-feedback faults. The number of faults is limited to randomly picked 10000 faults. The next columns show how often a fault has been detected (please note the modifications of the fault simulation algorithm). The column entitled 'Detections' gives the number of sure detections, i.e. there is a MIN/MAX difference at a primary output. The next column, only present for the five-valued logic, gives the number of 1/4 and 3/0 differences, i.e. the detection of the fault cannot be guaranteed. The column 'Detected' gives the number of *different* faults with MIN/MAX differences (fault coverage). Next, the simulation time is given. The logic distribution is shown in the following columns.

For normal logic operation Table 4 (three-valued logic, identical to that used in [15]) is to compare with Table 5 (five-valued logic). The following can be observed: For 34 of the 45 circuits, simulation based on the the five-valued logic results in (considerable) more MIN and MAX values than simulation based on the three-valued logic. The remaining 11 cases show no change. Comparing the results the five-valued based simulation achieves 7.98 million more MIN/MAX values than the three-valued based simulation. Accordingly, the number of

Circuit	Simulation No. faults	Detections	Detected	Time [s]	Logic Distribution 0	1	2
c0017	29	616	16	0.15	2081	960	2759
c0095	283	10793	205	2.76	100617	17501	79982
c0432	9132	79639	4322	138.11	2228434	300141	3863825
c0499	10000	22571	2208	397.55	15864926	647626	15487448
c0880	10000	85625	3255	233.69	12796368	494642	12708990
c1355	10000	17277	2919	549.58	15871118	610525	15518357
c1908	10000	38601	1956	585.07	12589533	516608	11893859
c2670	10000	79533	2435	579.99	68506804	489402	71003794
c3540	10000	38782	1776	559.05	11857797	731576	9410627
c5315	10000	37439	1565	594.38	67486312	501343	55012345
c6288	10000	42521	2682	4560.15	14379600	5596358	12024042
c7552	10000	36396	1665	659.49	47562753	582453	59854794
cs00027	66	1717	34	0.43	13552	2440	10408
cs00208	6377	43910	1685	51.97	2779578	245854	2713868
cs00298	8348	81244	4020	96.96	10820703	411681	5463616
cs00344	10000	90926	2541	192.45	12794938	574465	12630597
cs00349	10000	91258	2516	178.15	12798216	569316	12632468
cs00382	10000	110208	4943	153.12	18205107	505526	8289367
cs00386	10000	11970	790	96.19	12229016	259964	511020
cs00400	10000	106708	4929	146.35	18207127	503771	8289102
cs00420	10000	63444	2345	104.23	8543103	334141	8122756
cs00444	10000	95861	4625	155.73	18006504	492181	8501315
cs00510	10000	84498	3598	117.50	7379897	410360	5209743
cs00526	10000	98896	5124	133.72	18177157	429260	8393583
cs00641	10000	110969	2155	283.58	22217931	808432	19973637
cs00713	10000	100490	2063	301.62	22024314	775234	19200452
cs00820	10000	31934	4020	128.53	21935009	160406	1904585
cs00832	10000	31827	4140	147.67	21934401	160829	1904770
cs00838	10000	63864	2101	156.08	16903727	330392	15765881
cs00953	10000	116090	2995	226.37	37238089	583312	14178599
cs01196	10000	32115	2045	177.18	20048170	274335	11677495
cs01238	10000	31783	2037	176.76	20054411	272381	11673208
cs01423	10000	68445	2574	306.66	56048962	521547	22429491
cs01488	10000	4695	343	156.53	19743716	243651	5012633
cs01494	10000	4539	361	171.48	19742528	242611	5014861
cs05378	10000	93435	2923	751.59	137928587	742009	89329404
cs1196	10000	6516	315	218.84	20069319	252954	11677727
cs1269	10000	23455	788	498.30	20883070	599371	25517559
cs1512	7917	17083	668	291.76	43774205	300682	17677713
cs3330	10000	68133	1234	608.20	124931371	745970	79322659
cs3384	10000	72516	1158	779.62	117518877	1052831	90428292
cs344	10000	36306	1078	276.42	12767579	618714	12613707
cs4863	10000	57633	1158	808.38	70053944	1009983	48936073
cs6669	10000	107970	1736	1194.99	150737996	1259291	142002713
cs938	10000	62591	2085	156.34	16909336	325547	15765117
Sum	412152	2512822	100131	18103.67	1402666783	26508576	1009635241

Table 6. Results for delayed three-valued logic.

intermediate values using the five-valued logic is reduced by that number. This has a strong impact on the fault detection. Using the three-valued logic 2947268 times a fault could be detected. But due to the fault simulation based on the five-valued logic 3754805 (127.40%) *more* detection events are computed (6702073 in total). In addition, there are 1987851 potential detections. It might be possible, however, that the five-valued based fault simulation detects the very same subset of faults than the three-valued based fault simulation. But column 'Detected' clearly shows that this is not the case. The five-valued based simulation finds two times more faults than the other simulation. Please note, that here only sure detections are counted. Furthermore, the five-valued based simulation is faster than the three-valued based simulation.

In Table 6 and 7 the results are presented for the special logic exemplarily shown for the NOR operation in Table 3 on the right. This logic has been designed to not let a potential delay fault die due to the common logic operation. On the contrary, the logic provides the possibility that a delayed input signal, that might eventually cross the threshold, will be handed to the output of a gate, unless an other input takes a dominating value. For this set of experiments the same conclusions can be drawn as for the common logic discussed above. In addition, the number in the column of logic value 2 (five-valued logic) directly gives the the number of potential delay detections.

4 Conclusions

We presented a multiple-valued logic based bridging fault simulator that considers a user defined mapping of voltages to logic values to take more advantage in the determined bridge values and, more importantly, to more accurately regard manufacturing variances.

To determine the bridge logic value, the proposed simulator considers only the current input, the topology and type of the active transistor networks. No other simulation is performed during run time, and no lookup in tables of large size must be done. This results in a small and fast bridging fault simulator. Furthermore, it is able to note possible timing conflicts and possible uncertain output voltages by adequately processing intermediate logic values.

Besides the more distinct logic value distribution, the experimental results show that the behavior of the circuit is more accurately determined due to the considerably less number of intermediate values. Furthermore, since this comes with more MIN and MAX values, the quality of a potential test sequence can be determined more precisely. Furthermore, besides the large number of potential detection the (sure) fault detection is enhanced by 127% compared to the three-valued based simulation. All this improves the quality and precision of the bridging fault simulation.

The next steps of this work are as follows: The delay logic will be improved by taking into consideration the actual transition direction to make more precise pre-

	Simulation					Logic Distribution				
Circuit	No. faults	Detections	Prob. Det.	Detected	Time [s]	0	1	2	3	4
c0017	29	668	133	18	0.14	2202	59	613	129	2797
c0095	283	11492	1244	221	2.78	101208	1537	12475	1483	81397
c0432	9132	123280	6467	7630	127.39	2271192	14269	168021	8543	3930375
c0499	10000	22571	57337	2208	383.90	15864926	80545	446398	120683	15487448
c0880	10000	149325	14604	6252	198.43	12841445	24271	332267	19502	12782515
c1355	10000	91530	12933	8209	409.57	16020356	44	288208	18561	15672831
c1908	10000	128281	7062	5665	506.34	12668347	95	354245	9389	11967924
c2670	10000	110270	11133	4252	559.76	68532609	6753	413039	18286	71029313
c3540	10000	94188	5567	3703	500.38	11912714	3281	613214	9422	9461369
c5315	10000	63495	12075	3896	589.61	67508621	9601	429132	20205	55032441
c6288	10000	745165	5874	9659	1419.08	16900075	8703	716864	204	14374154
c7552	10000	94702	9786	5188	600.78	47619428	5958	443305	18873	59912436
cs00027	66	1824	174	41	0.44	13624	265	1867	36	10608
cs00208	6377	66369	15275	2803	56.30	2801768	10658	181297	12506	2733071
cs00298	8348	102937	12977	4580	95.36	10841675	25161	341955	4793	5482416
cs00344	10000	138989	32668	4889	188.35	12858551	30215	400902	27382	12682950
cs00349	10000	138474	32616	4944	176.78	12860978	31868	397001	25663	12684490
cs00382	10000	153877	24623	6850	159.57	18257432	40712	358085	16097	8327674
cs00386	10000	11970	0	790	92.90	12229016	0	259964	0	511020
cs00400	10000	149347	25432	6995	141.77	18259513	42525	353537	17628	8326797
cs00420	10000	91629	23546	3584	102.71	8569916	18783	244084	19435	8147782
cs00444	10000	143814	20224	7354	147.24	18062304	36011	338548	16252	8546885
cs00510	10000	161902	14218	7379	111.72	7462211	10712	234929	14678	5277470
cs00526	10000	126860	18420	5884	138.71	18205741	24174	337911	11673	8420501
cs00641	10000	110969	6343	2155	274.53	22218716	4395	795306	7681	19973902
cs00713	10000	102930	12267	2240	290.26	22029757	5717	746476	13147	19204903
cs00820	10000	43492	10589	5380	128.82	21946611	24340	104849	8906	1915294
cs00832	10000	43707	11144	5470	133.58	21946428	23620	104431	9740	1915781
cs00838	10000	91757	22540	3211	153.30	16926336	18628	243759	19486	15791791
cs00953	10000	233408	21982	5976	209.79	37395060	12255	278870	21181	14292634
cs01196	10000	52328	21330	3831	184.26	20078192	17061	196080	14667	11694000
cs01238	10000	53572	24141	3963	163.63	20085210	20410	187657	15517	11691206
cs01423	10000	85296	26819	3899	289.37	56063445	38012	423661	22808	22452074
cs01488	10000	4695	0	343	152.12	19743716	0	243651	0	5012633
cs01494	10000	4539	0	361	153.29	19742528	0	242611	0	5014861
cs05378	10000	145786	28298	4410	742.73	137976666	38965	596043	14613	89373713
cs1196	10000	6516	0	315	224.01	20069319	0	252954	0	11677727
cs1269	10000	23455	0	788	508.06	20883070	0	599371	0	25517559
cs1512	7917	17083	0	668	284.67	43774205	0	300682	0	17677713
cs3330	10000	68177	3944	1243	609.95	124932237	2276	739250	3555	79322682
cs3384	10000	72516	0	1158	715.63	117518877	0	1052831	0	90428292
cs344	10000	36306	0	1078	265.03	12767579	0	618714	0	12613707
cs4863	10000	57633	0	1158	897.20	70053944	0	1009983	0	48936073
cs6669	10000	107970	0	1736	1145.22	150737996	0	1259291	0	142002713
cs938	10000	89964	21693	3116	148.14	16931346	19236	241926	18680	15788812
Sum	412152	4375058	575478	165493	14383.60	1406487090	651115	17906257	581404	1013184734

Table 7. Results for delayed five-valued logic.

dictions. Furthermore, sequential circuits and bridging faults causing feedbacks will be considered.

References

[1] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[2] J.K. Acken and S.D. Millman. Fault model evolution for diagnosis; accuracy vs precision. In *Custom Integrated Circuits Conference*, pages 13.4.1–13.4.4, 1992.

[3] Prithviraj Banerjee and Jacob A. Abraham. A multivalued algebra for modeling physical failures in MOS VLSI circuits. *IEEE Trans. on CAD*, 4(5):312–321, 1985.

[4] B.Chess and C. Roth. On evaluating competing bridge fault models for CMOS ICs. In *VLSI Test Symp.*, pages 446–451, 1994.

[5] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.

[6] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational circuits and a target translator in fortran. In *Int'l Symp. Circ. and Systems, Special Sess. on ATPG and Fault Simulation*, pages 663–698, 1985.

[7] A. Chatzigeorigou, S. Nikolaidis, and I. Tsoukalas. A modeling technique for CMOS gates. *IEEE Trans. on CAD*, 18(5):557–575, 1999.

[8] T. Chen and I.N. Hajj. GOLDENGATE: fast and accurate bridging fault simulator under a hybrid logic/i_{DDQ} testing environment. In *Int'l Conf. on CAD*, pages 555–561, 1997.

[9] Chennian Di and Jochen A.G. Jess. An efficient CMOS bridging fault simulator: With SPICE accuracy. *IEEE Trans. on CAD*, 15(9):1071–1080, 1996.

[10] Michele Favalli, P. Olivo, and Bruno Riccò. A probabilistic fault model for "analog" faults in digital CMOS circuits. *IEEE Trans. on CAD*, 11(11):1459–1462, 1992.

[11] F.J. Ferguson and T. Larrabee. Test pattern generation for realistic bridge fault in CMOS ICs. In *Int'l Test Conf.*, pages 492–499, 1991.

[12] G.S. Greenstein and J.H. Patel. E-PROOFS: a CMOS bridging fault simulator. In *Int'l Conf. on CAD*, pages 268–271, 1992.

[13] T. Lee, W. Chuang, I.N. Hajj, and W.K. Fuchs. Circuit-level dictionaries of CMOS bridging faults. In *VLSI Test Symp.*, pages 386–391, 1994.

[14] P.C. Maxwell and R.C. Aitken. Biased voting: A method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In *Int'l Test Conf.*, pages 63–72, 1993.

[15] W. Meyer and R. Camposano. Active timing multilevel fault-simulation with switch-level accuracy. *IEEE Trans. on CAD*, 14(10):1241–1256, 1995.

[16] S.D. Millman and Sir J.P. Garvey. An accurate bridging fault test pattern generator. In *Int'l Test Conf.*, pages 411–418, 1991.

[17] S.D. Millman, E.J. McCluskey, and J.K. Acken. Diagnosing CMOS bridging faults with stuck-at fault dictionaries. In *Int'l Test Conf.*, pages 860–870, 1990.

[18] J. Rearick and J.H. Patel. Fast and accurate CMOS bridging fault simulation. In *Int'l Test Conf.*, pages 54–62, 1993.

[19] M. Renovell, P. Huc, and Y. Bertrand. CMOS bridge fault modeling. In *VLSI Test Symp.*, pages 392–397, 1994.

[20] S. Sengupta, S. Kundu, S. Chakravarty, P. Paravathala, R. Galivanche, G. Kosonocky, M. Rodgers, and TM Mak. Defect-based test: A key enabler for successful migration to structural test. *Intel Technology Journal*, 1, 1999.