Approximate Computing

Salman Ahmed

Motivation and Ad-hoc Approaches

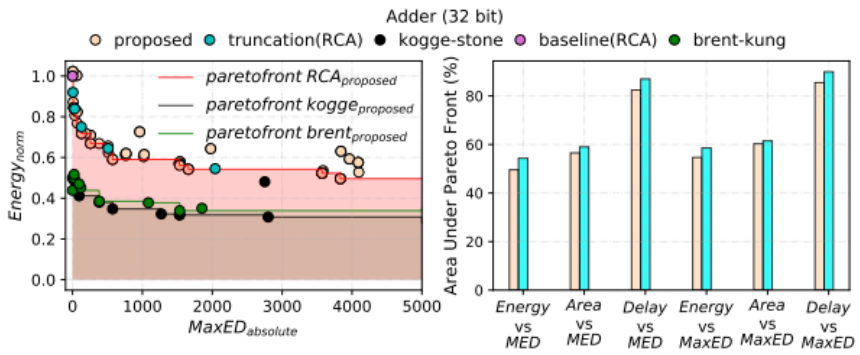
- Declining Moore's law
- Dark Silicon
- Noisy input
- Limited data precision
- Physical defects
- Additional load or hard real-time constraints
- Perceptual limitations of humans
- Trade off quality with efficiency

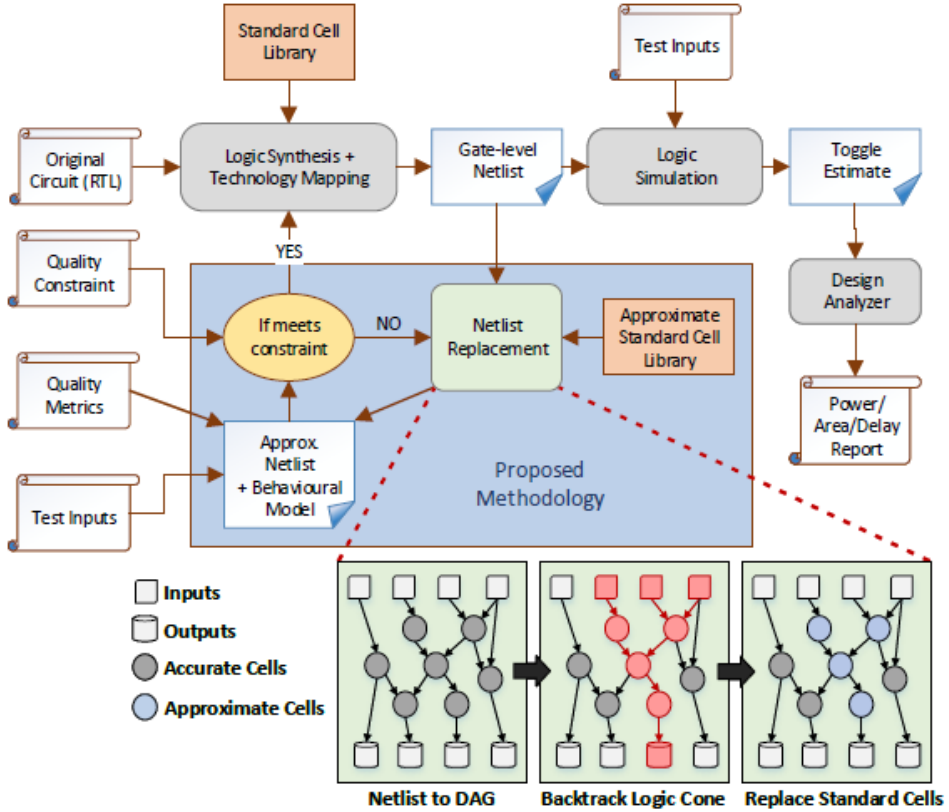
- Precision Scaling
- Loop Perforation
- Load Value Approximation
- Memoization
- Skipping Tasks and Memory Accesses
- Using Multiple Inexact Program Versions
- Using Inexact or Faulty Hardware
- Using Voltage Scaling
- Approximating DRAM Memories

S. Mittal "A Survey of Techniques for Approximate Computing", in ACM Computing Surveys, Vol. 48, No. 4, Article 62, Publication date: March 2016.

ALS: Using Approximate Std. Cell Library

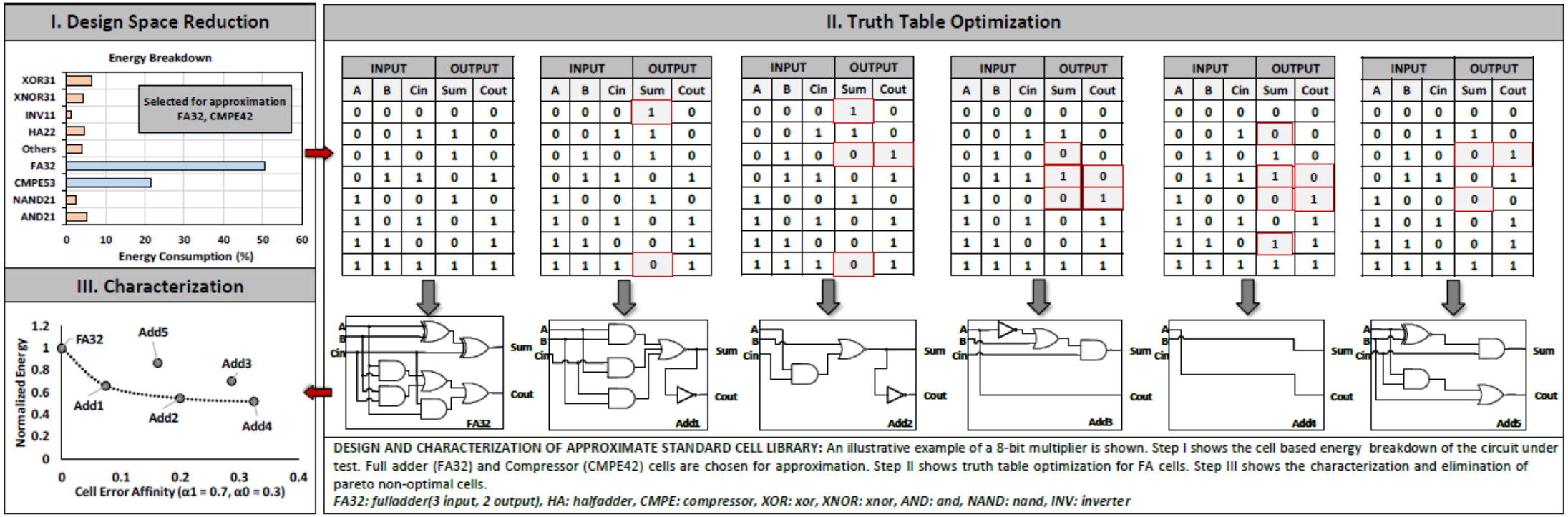
- Design Space Reduction
- Truth table Optimization
- Characterization





S. De, J. Huisken and H. Corporaal, "An Automated Approximation Methodology for Arithmetic Circuits," *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019.

ALS: Using Approximate Std. Cell Library



DESIGN AND CHARACTERIZATION OF APPROXIMATE STANDARD CELL LIBRARY: An illustrative example of a 8-bit multiplier is shown. Step I shows the cell based energy breakdown of the circuit under test. Full adder (FA32) and Compressor (CMPE42) cells are chosen for approximation. Step II shows truth table optimization for FA cells. Step III shows the characterization and elimination of pareto non-optimal cells.

FA32: fulladder(3 input, 2 output), HA: halfadder, CMPE: compressor, XOR: xor, XNOR: xnor, AND: and, NAND: nand, INV: inverter

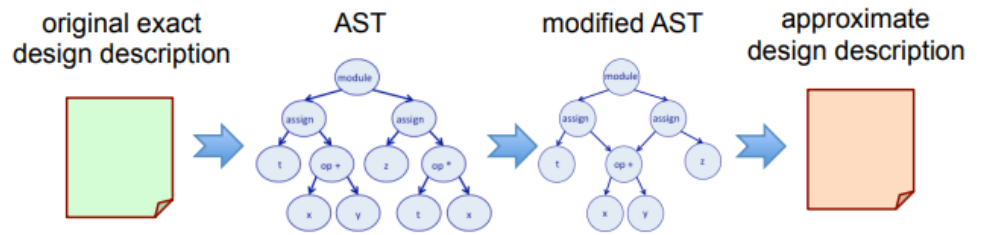
S. De, J. Huisken and H. Corporaal, "An Automated Approximation Methodology for Arithmetic Circuits," *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019.

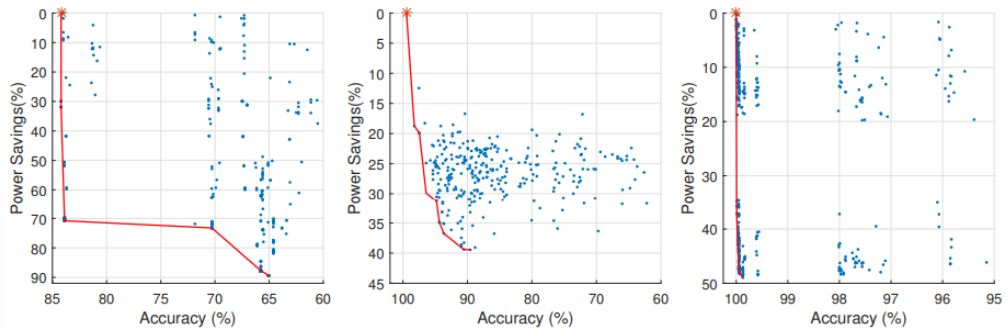
ALS: Using AST Transformations

- Data Type Simplifications
- Operation Transformations
- Arithmetic Expression Transformations
- Variable-to-Constant Substitution Transformations
- Loop Transformations

```
always @(*)
begin
 sum_temp[0]=result[0];
 for (i=1; i<nSVs; i=i+1)
 begin
  sum_temp[i] = sum_temp[i-1] + result[i];
 end
end
```

```
always @(*)
begin
 sum_temp[0]=(sum_temp[2] | result[3]);
 sum_temp[1]=(sum_temp[0] | result[1]);
 sum_temp[2]=(sum_temp[1] | result[2]);
 sum_temp[3]=({result[0][15:3], {3'b000}});
 sum_temp[4]=(sum_temp[3] + result[4]);
 sum_temp[5]=(sum_temp[4] + result[5]);
end
```

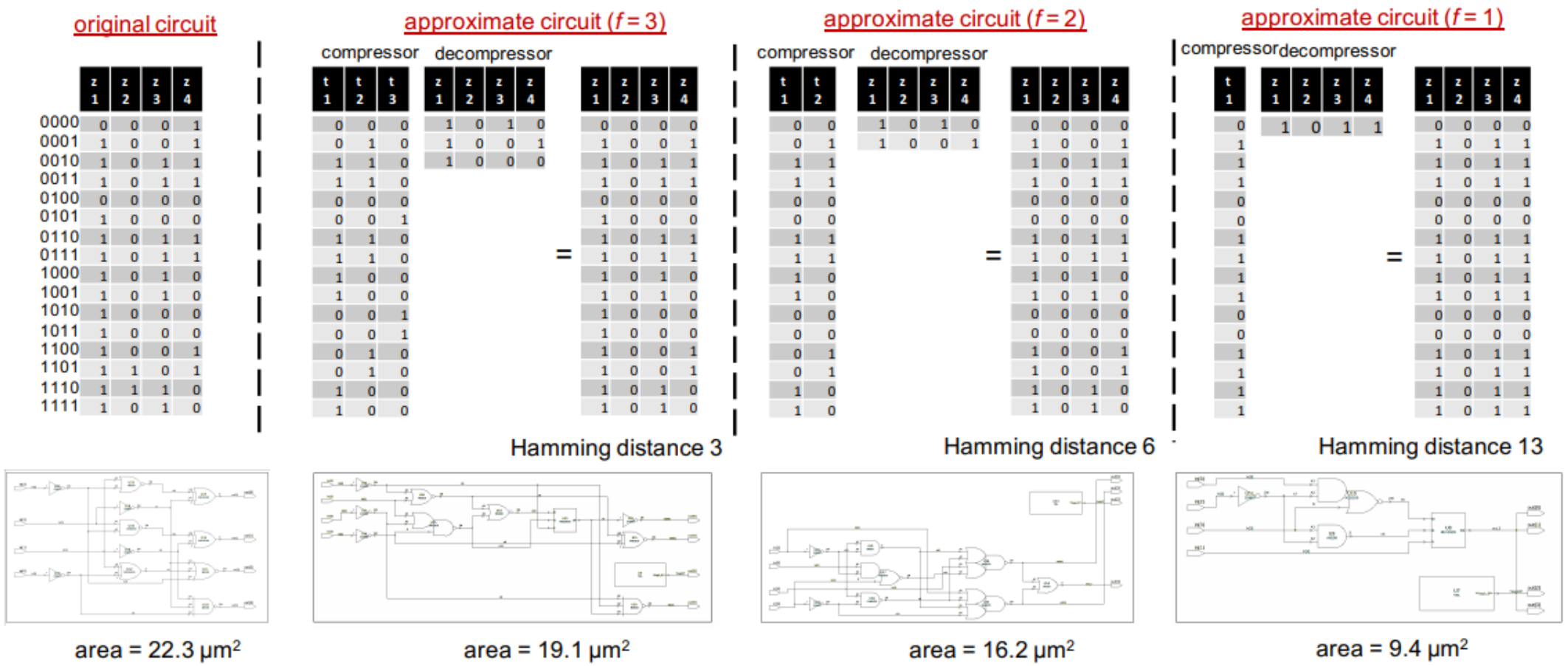




(a) Perceptron (b) FIR Filter (c) FFT

K. Nepal, S. Hashemi, H. Tann, R. I. Bahar and S. Reda, "Automated High-Level Generation of Low-Power Approximate Computing Circuits," in *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 18-30, 1 Jan.-March 2019.

ALS: Using Boolean Matrix Factorization



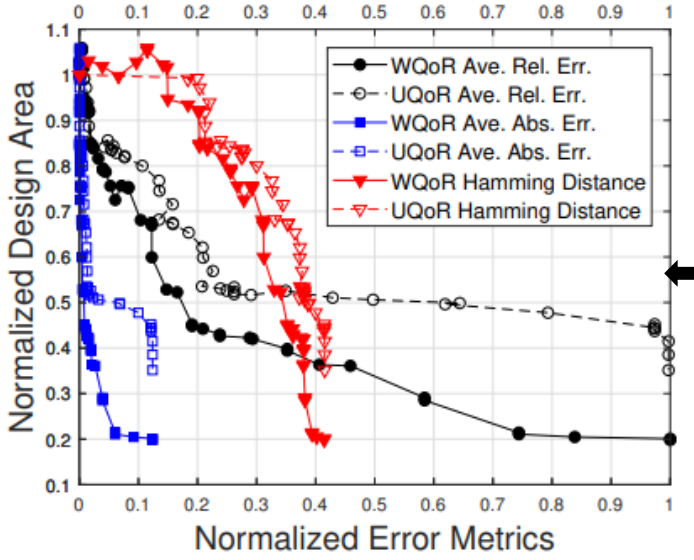
S. Hashemi, H. Tann and S. Reda, "BLASYS: Approximate Logic Synthesis Using Boolean Matrix Factorization," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1-6.

ALS: Using Boolean Matrix Factorization

Algorithm 1: BLASYS: Boolean Level Approximate Circuit Synthesis

Input : Accurate Circuit $ACir$, Error Threshold
Output: Approximate Circuit Cir

```
subcircuits=Decompose input circuit using k × m decomposition
// Factorization profiling Phase
for each subcircuit s_i with m_i ≤ m outputs do
    M=Construct truth table of s_i
    // profile for every possible factorization degree
    for f=1 to m_i-1 do
        [B, C] = BMF(M, f)
        T_{s_i,f}=Construct truth table of BC
    end
end
// Circuit Space Exploration Phase
Cir=ACir;
Let f_i = m_i for all subcircuits s_i
while QoR(Cir) < threshold do
    for each subcircuit s_i with f_i > 1 do
        Cir'=Cir(s_i → T_{s_i,f_i-1})
        Δerr_i = QoR(Cir') − QoR(Cir)
    end
    b = arg min_i(Δerr_i)
    Cir = Cir(s_b → T_{s_b,f_b-1})
    f_b = f_b − 1
end
Cir=Synthesize Best new Design
return Cir
```



Constraint: Minimization of $\|(M - S_1.S_2)w\|_2$ for target QoR

Design	Area Savings (%)	Power Savings (%)	Delay Reduction (%)
Adder32	44.78	63.79	12.07
Mult8	28.77	26.87	12.32
BUT	7.87	11.25	2.23
MAC	47.55	55.58	64.41
SAD	32.80	41.47	69.14
FIR	19.52	22.26	12.18

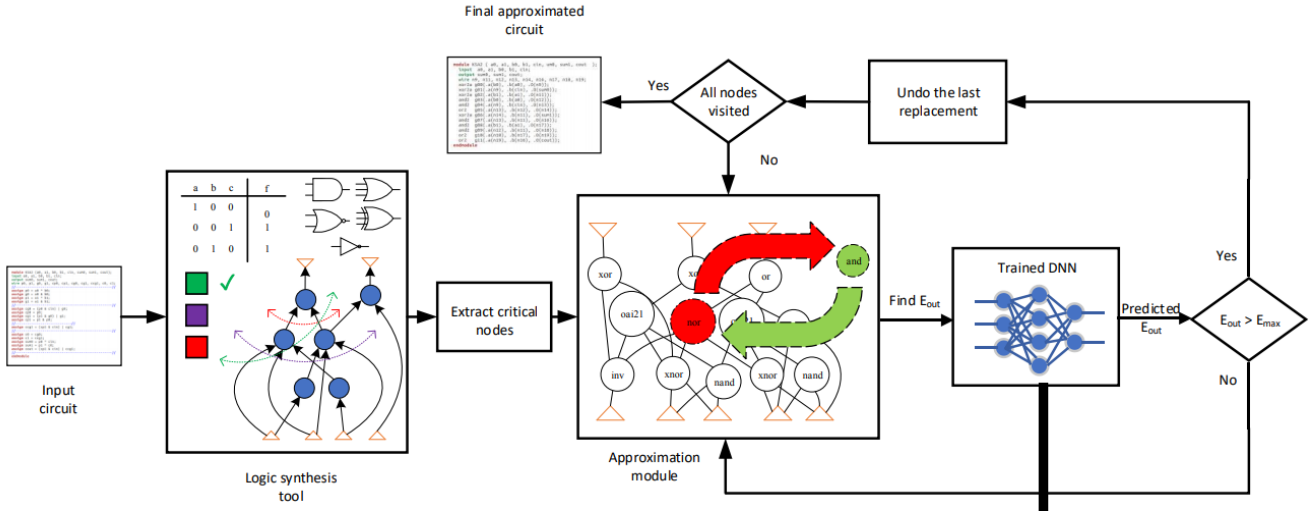
S. Hashemi, H. Tann and S. Reda, "BLASYS: Approximate Logic Synthesis Using Boolean Matrix Factorization," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1-6.

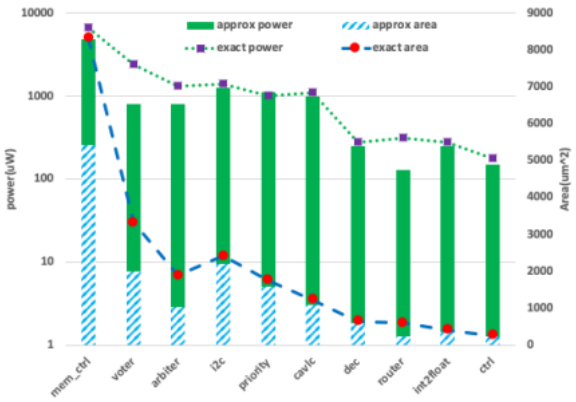
ALS: Using Pre-trained Error Network

Algorithm 2: Deep-PowerX Inference

Input: Input circuit $G = (V, E)$, Technology library, Max. error constraint (E_{max})
Output: Approx. circuit $G_X = (V_X, E_X)$

// Initialization:
1 G_X = Copy G.
2 V' = Extract top 20% of active nodes of V_X.
3 V'' = Extract top 20% immediate fanouts of nodes in V'.
4 set E_{out} to 0.0, and copy technology library to *Lib*.
// Minimizing power:
5 Lib1 (Lib2) = Sort Lib in ascending order by gates' out (in) cap.
6 opt_mode = power;
7 $V''' = V' \cup V''$
8 **for** *each node n in* V''' **do**
9 **if** *opt_mode is power* **then**
10 **if** $n \in V'$ **then**
11 Lib = Lib1;
12 **else if** $n \in V''$ **then**
13 Lib = Lib2;
14 **for** *each gate g in Lib* **do**
15 Replace n with g; Utilize DNN to infer E_{pred}
16 **if** *replacement is accepted* **then**
17 Update E_{out} with E_{pred}.
18 Break;
19 **else**
20 Undo the last gate replacement.
21 **if** $E_{out} > E_{max}$ **then**
22 Undo the last gate replacement.
23 Break;
// Minimizing area:
24 **if** $E_{out} < E_{max}$ **then**
25 Lib = Sort Lib in ascending order by gates' area.
26 opt_mode = area;
// Copy nodes of approx. circuit in power minimization phase:
27 $V''' = V_X$; Go to line 8.
28 **return** G_X





Network structure	93-400-300-51
Learning rate	0.001
Optimizer	Adam
Loss function	Binary cross entropy
Epochs	30

2020. Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design. Association for Computing Machinery, New York, NY, USA.

ALS: Using LUT Memorization

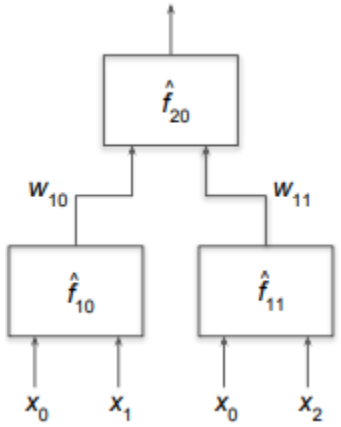
$$\hat{f} : \mathbb{B}^k \to \mathbb{B} \qquad b \in \mathbb{B}$$

$$\hat{f}(p) = \begin{cases} 1 & \text{if } c_{p1} > c_{p0}, \\ 0 & \text{if } c_{p1} < c_{p0}, \\ b & \text{if } c_{p1} = c_{p0} \end{cases}$$

x / $x_0x_1x_2$	y
000	0
000	1
000	1
001	1
100	0
110	0
110	1

p / $x_0x_1x_2$	y^0	y^1
000	1	2
001	0	1
010	0	0
011	0	0
100	1	0
101	0	0
110	1	1
111	0	0

p	$\hat{f}$
000	1
001	1
010	0^*
011	1^*
100	0
101	1^*
110	1^*
111	0^*



p / x_0x_1	y^0	y^1	$\hat{f}_{10}$
00	1	3	1
01	0	0	1^*
10	1	0	0
11	1	1	1^*

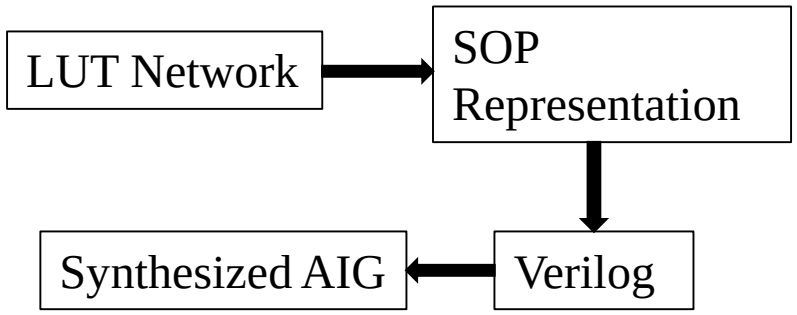
p / x_0x_2	y^0	y^1	$\hat{f}_{11}$
00	1	2	1
01	0	1	1
10	2	1	0
11	0	0	1^*

x / $x_0x_1x_2$	$w_{10}w_{11}$	y
000	11	0
000	11	1
000	11	1
001	11	1
100	00	0
110	10	0
110	10	1

p / $w_{10}w_{11}$	y^0	y^1	$\hat{f}_{20}$
00	1	0	0
01	0	0	1^*
10	1	1	0^*
11	1	3	1

S. Chatterjee. “Learning and memorization”. Proceedings of the 35th International Conference on Machine Learning 2018

ALS: Using LUT Memorization

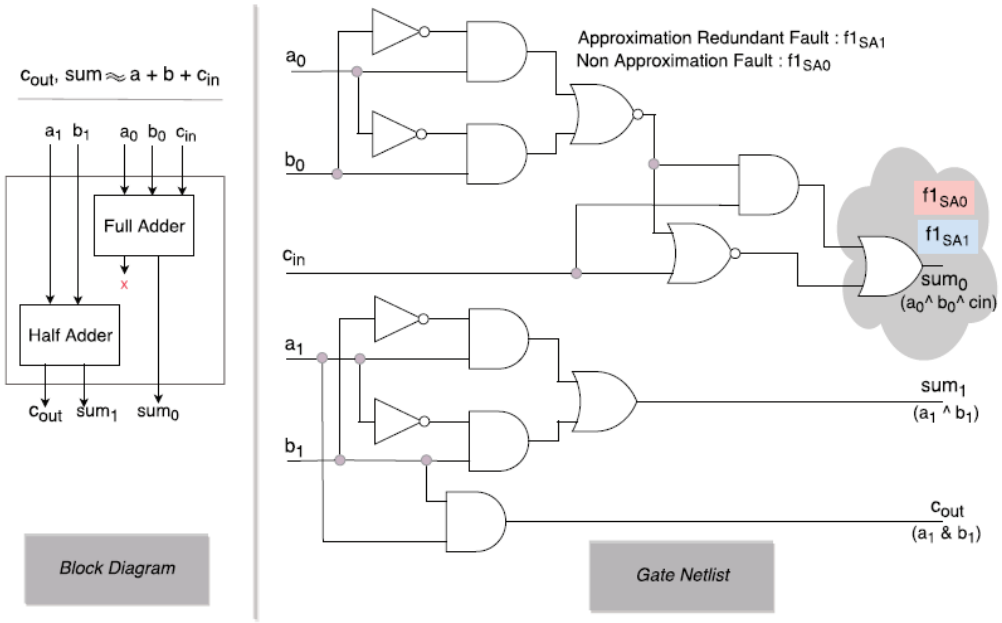


LAYER	LUT COUNT	TRAINING ACCURACY			
		MEAN	STD	MIN	MAX
0	784	0.5072	0.0340	0.4042	0.6572
1	1024	0.6055	0.0403	0.5120	0.7299
2	1024	0.7431	0.0191	0.6721	0.7877
3	1024	0.8297	0.0068	0.8038	0.8526
4	1024	0.8655	0.0033	0.8562	0.8751
5	1024	0.8808	0.0015	0.8759	0.8853
6	1	0.8898	0.0000	0.8898	0.8898

	PLANE	AUTO	BIRD	CAT	DEER	DOG	FROG	HORSE	SHIP	TRUCK
PLANE		0.77	0.77	0.80	0.82	0.81	0.84	0.81	0.71	0.79
AUTO	0.96		0.79	0.77	0.79	0.80	0.80	0.78	0.76	0.67
BIRD	0.95	0.98		0.68	0.63	0.69	0.66	0.72	0.83	0.81
CAT	0.96	0.98	0.96		0.70	0.61	0.68	0.71	0.81	0.76
DEER	0.96	0.98	0.95	0.96		0.73	0.69	0.71	0.83	0.81
DOG	0.98	0.99	0.97	0.96	0.97		0.72	0.70	0.82	0.79
FROG	0.97	0.98	0.95	0.95	0.97	0.96		0.75	0.85	0.80
HORSE	0.98	0.99	0.96	0.97	0.96	0.98	0.97		0.81	0.75
SHIP	0.93	0.96	0.98	0.98	0.98	0.98	0.99	0.98		0.77
TRUCK	0.97	0.95	0.98	0.97	0.97	0.98	0.98	0.97	0.98	

S. Chatterjee. “Learning and memorization”. Proceedings of the 35th International Conference on Machine Learning 2018

Test of Approximate Circuits



ISCAS circuits[4]	#gates	f_{orig}	f^{bf}_{final}	f^{bf}_{Δ} (%)	sec
c499*	577	1320	1153	12.65%	73s
c880*	527	1074	305	71.60%	31s
c1355*	575	1330	1196	10.08%	79s
c1908*	427	974	949	2.57%	30s
c2670*	931	1950	428	78.05%	396s
c3540*	1192	2657	839	68.42%	418s
c5315*	2063	4224	1648	60.98%	6224s
c6288*	2836	7048	3071	56.42%	4881s

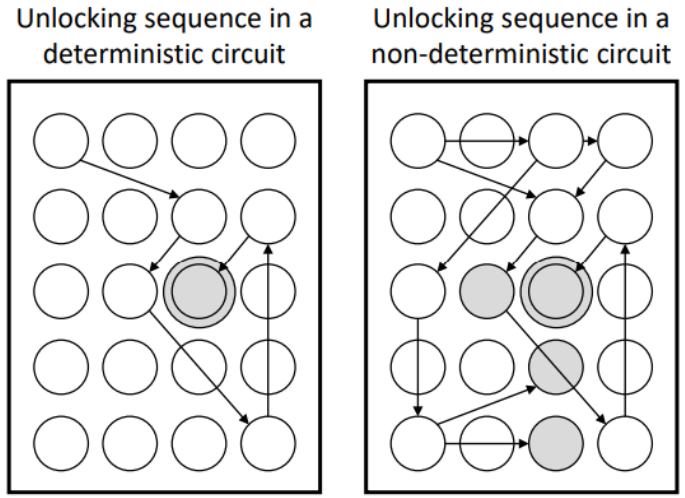
	Correct†		Appx‡		Appx:SA0⋆		Appx:SA1±	
	In	Out†	Out‡	$e^{\ddagger}$	Out⋆	$e^{\star}$	Out±	$e^{\pm}$
	00000	000	000	0	000	0	001	1
	00001	001	001	0	000	1	001	0
	00010	010	010	0	010	0	011	1
	00011	011	011	0	010	1	011	0
	00100	001	001	0	000	1	001	0
	00101	010	000	2	000	2	001	1
	00110	011	011	0	010	1	011	0
	00111	100	010	2	010	2	011	1
	01000	010	010	0	010	0	011	1
	01001	011	011	0	010	1	011	0
	01010	100	100	0	100	0	101	1
	01011	101	101	0	100	1	101	0
	01100	011	011	0	010	1	011	0
	01101	100	010	2	010	2	011	1
	01110	101	101	0	100	1	101	0
	01111	110	100	2	100	2	101	1
	10000	001	001	0	000	1	001	0
	10001	010	000	2	000	2	001	1
	10010	011	011	0	010	1	011	0
	10011	100	010	2	010	2	011	1
	10100	010	000	2	000	2	001	1
	10101	011	001	2	000	3	001	2
	10110	100	010	2	010	2	011	1
	10111	101	011	2	010	3	011	2
	11000	011	011	0	010	1	011	0
	11001	100	010	2	010	2	011	1
	11010	101	101	0	100	1	101	0
	11011	110	100	2	100	2	101	1
	11100	100	010	2	010	2	011	1
	11101	101	011	2	010	2	011	2
	11110	110	100	2	100	2	101	1
	11111	111	101	2	100	3	101	2

†, ‡ Golden non-approx, approx (carry cut) 2-bit adder responses
⋆, ± Approx adder with SA0, SA1 at sum_0 ($f1_{SA0}$, $f1_{SA1}$)
In:$C_{in}\,a_1\,a_0\,b_1\,b_0$, Out:$C_{out}\,sum_1\,sum_0$
e: error in each case, worst-case errors $wc^{\ddagger}$=2,$wc^{\star}$=3, $wc^{\pm}$=2

A. Chandrasekharan, S. Eggersglüß, D. Große and R. Drechsler, "Approximation-aware testing for approximate circuits," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018.

Insecurity of Approximate Circuits

Attack model	Key features	
No. 1: tamper interconnect	Assumption	(1) AC functional IP is a blackbox; (2) Interconnect between AC and non-AC IPs are accessible;
	Attack method	(1) Swap MSB and LSB bits; (2) Force LSB to stuck-at-0/1;
No. 2: tamper AC function	Assumption	(1) AC functional IP is a whitebox; (2) Non-AC IPs are protected blackbox;
	Attack method	(1) Use hardware Trojan to trigger malicious approximate function; (2) Use external control to alter ambient environment.



F. Regazzoni, C. Alippi and I. Polian, "Security: The Dark Side of Approximate Computing?," *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1-6.
Pruthvy Yellu, Mezanur Rahman Monjur, Timothy Kammerer, Dongpeng Xu, Qiaoyan Yu, "Security Threats and Countermeasures for Approximate Arithmetic Computing", *Design Automation Conference (ASP-DAC) 2020 25th Asia and South Pacific*, pp. 259-264, 2020.