International Journal of Gender, Science and Technology
..ISSN: 2040-0748

UGC Care Group I Journal
Vol-10 Issue-02 Sep 2021

Development of an Energy-Efficient Low-Latency Signed Multiplier for Accelerating Hardware in FPGAs.

Dr. Subrat Kumar Mohanty
Department of Electronics and Telecommunication, College of Engineering Bhubaneswar

***Abstract*— Multiplication is one of the most commonly usedarithmeticoperationsinawiderange ofapplications,suchasmultimediaprocessingandartificialneuralnet works.Multiplier is one of the main causes of energy consumption, critical path latency, and resource utilization for these kinds of applications.In designs based on field-programmable gate arrays (FPGAs), these effects become more noticeable. Still, the majority of cutting-edge designs were created for ASIC-based systems.Moreover, the majority of existing field-programmable gate array (FPGA) designs are restricted to unsigned integers and necessitate additional circuits to accommodate signed operations. To address these limitationsfortheFPGA-basedimplementationsofapplicationsutilizingsignednumbers,thisl etterpresentsanarea-optimized,low-latency, and energy-efficient design for an accurate signedmultiplier.Our implementations give an abrupt 40.0%, 43.0%, and 70.0% reduction in area, latency, and energy, respectively, as compared to the Vivada area-optimized multiplier IP.**

***IndexTerms*— Acceleratorarchitectures,artificialneuralnetworks(ANN),fixed-pointarithmetic,field-programmablegatearrays(FPGAs),multiplyingcircuits.**

I. Introduction

APPLICATIONS in the domain of digital signal process-ingandmachinelearningextensivelyusemultiplication asoneofthebasicarithmeticoperations.Thearchitectureof a selected multiplier and its implementation directly affectthe overall performance, resource utilization, and energy con-sumption of such applications. The FPGA synthesis tools tendto use DSP blocks for high-performance multiplication [1].However, two points are worth noting concerning the DSPblocksutilization.

1) For many applications, such as artificial neural networks(ANNs), the 32-b floating-point precision is often notnecessary for obtaining acceptable quality results. Asdiscussed in Section III, our 8-b quantized implementa-tion of an ANN reduces the classification accuracy onlyby 0.42% when compared with full-precision classifi-cationaccuracy.Forimplementingmultipliersforthese low-precision numbers, the synthesis tools opt to uselookuptables(LUTs)insteadofDSPblocks.
2) As noted by Ullah *et al.* [2] and Kuon and Rose [3], due tothe nonuniform distribution of these DSP blocks acrossthe FPGA, the critical path delay could be adverselyaffected when many of them have to be concatenated forlarge multiplication operations. Moreover, DSP resourcesare limited. On the other hand, the LUT resources aremuch larger. They also offer comparable performancewithbetterenergy-efficiencyandflexibilitythantheDSPblocksforsmall-sizedmultipliers.Therefore,itismoreadvantageoustohavet heoptiontousethelow-area, high performance, and energy-efficient LUT-basedmultiplierbesidetheDSPblocks.Inthisletter,weprovi de area-optimized, low-latency, and energy-efficientaccuratesignedmultipliersforFPGA-basedsystems.

FPGA vendors, such as Xilinx and Intel, provide softcoreLUT-basedmultipliers(signedandunsigned)asdescribedin [4]. These multipliers can be either area or speed optimized.Booth's algorithm [5] is also a commonly used technique formultiplicationbecauseitreducesthetotalnumberofgeneratedp artial products by encoding the multiplier bits. The widelyknownrelated worksare [7]–[9] and [11]. Kumm*et al.* [7]andWalters[8]haveusedBooth'salgorithmtopresentarea-efficient radix-4 multiplier implementations for XilinxFPGAs. However, these implementations do not use compres-sor trees for adding the generated partial products and havelarge critical path delays. More importantly, Kumm *et al.* [7]has not discussed the implementations for signed numbers.Parandeh-Afshar *et al.* [11] have proposed a partial productcompressor tree for Altera (now Intel) FPGAs. Nonetheless,their generalized parallel counters underutilize LUTs in twoconsecutiveadaptivelogicmodules(ALMs).Theirfollow-upwork,Parandeh-AfsharandIenne[9]haveusedtheBooth'sandBaugh-Wooley'smultiplication[6]algorithmsforarea-efficientmultiplierimplementation.However,inordertoreduceth eeffectivelengthofthecarrychains,theirdesign limits the length of the ALM to five, resulting in theunderutilizationoftheFPGAresources.

On the other hand, Kakacak [12] and Kumm *et al.* [13]utilized smaller multiplier blocks for designing higher ordermultipliers. However, such techniques prove to be only usefulforsmallbit-widthmultipliers;forhigherbit-widthmultipliers,they consume more FPGA resources. For example, the logic-basedimplementation(usingthe "*" operation) of anaccurate 8×8 multiplier on Virtex-7 FPGA in Xilinx Vivado, withdefaultsynthesisoptions,consumes71LUTs,whereasthe modularimplementationofanaccurate8×8multiplierusingaccurate4×4 multipliersconsumes82LUTs.

A. MotivationforSignedMultipliers

ManuscriptreceivedMarch10,2020;revisedApril30,2020;accepted May10,2020.DateofpublicationMay15,2020;dateofcurrentver-sionMay27,2021.Thisworkwassupportedbythe GermanResearchFoundation(DFG)fundedProjectReApunderGrant380524764.Thismanuscriptwasrecommended forpublicationbyJ.Hu.*(Correspondingauthor:AkashKumar.)*

SalimUllahandAkashKumarawiththeDepartmentofProcessorDesign, Technische Universität Dresden, 01062 Dresden, Germany (e-mail:salim.ullah@tu-dresden.de;akash.kumar@tu-dresden.de).

Tuan Duy Anh Nguyen was with the Department of Processor Design,Technische Universität Dresden, 01062 Dresden, Germany (e-mail: duyat-uan@acm.org).

TABLE I
BOOTH ENCODING AND CORRESPONDING SE FOR PARTIAL PRODUCTS. b_{m+1}, b_m, AND b_{m-1} ARE MULTIPLIER BITS. (a) RADIX-4 BOOTH ENCODING. (b) SE

(a)

(b)

utilizing unsigned multiplier designs. For example, we have quantized the trained parameters (weights and biases) of a lightweight ANN to 8-bfixed-point numbers to implement the ANN on FPGA. These parameters are signed numbers. To implement the ANN hardware using unsigned multipliers, we require additional signed-unsigned converters to extract the sign bit from the operands and compute the final product sign. These converters receive 2's complement numbers and produce corresponding numbers in sign–magnitude format. After multiplication in sign–magnitude format, the result is converted back to the 2's complement scheme using signed–unsigned converter. These additional modules have increased the critical path delay of each multiplier by 2.061 ns and LUTs utilization by 24. Therefore, for the hardware implementations of appli-cations utilizing signed numbers, it is always advantageous to have high performance signed arithmetic units.

B. Novel Contributions

Our contributions include the following.

1) *A Novel Architecture for Booth Multiplier:* Using 6-input LUTs and associated fast carry chains of modern FPGAs, we present an architecture for signed multipliers that provides better performance[1] than state-of-the-art designs.
2) *Parallel Generation of Partial Products:* We eliminate the need for sequential computation of the partial prod-ucts and generate all Booth-encoded partial products in parallel; that significantly reduces the overall critical path delay of the multiplier.
3) *Efficient Partial Products Encoding:* Our partial product encoding technique reduces the length of the carry chain in each partial product to further reduce the critical path of the multiplier.

II. PROPOSED DESIGNS OF ACCURATE MULTIPLIERS

Using the concepts of radix-4 Booth's multiplication algorithm, we present our area-optimized, low-latency, and energy-efficient accurate signed multipliers. The correct sign of a partial product, in booth's encoding (BE)-based multiplier, is decided by the sign of the multiplicand (the MSB) and the corresponding value of BE. Table I(a) and (b) shows the list of required sign extensions (SEs) for all possible combinations of BE's values and MSB of the multiplicand. We have used Bewick's SE technique [16] to implement the correct sign of a partial product. Unlike state-of-the-art implementations, our proposed architecture computes all partial products in parallel and then adds the generated partial products using multiple 4:2 compressors and a ripple carry adder (RCA). The parallel generation of partial products significantly reduces the critical

[1] Collective performance considering the area, delay, and energy.

may still be possible to implement the required hardware acceleer ors

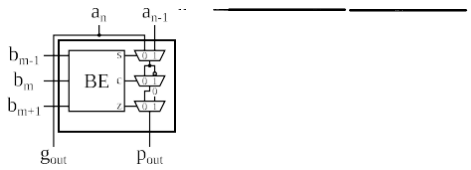

Fig.1. ConfigurationofLUTsusedinproposeddesign.(a)Type-*A*. (b) Type-*B*.(c)Type-*C*.

pathdelayofthemultiplier.Ourimplementationsprovideoptimized configurations for the 6-input LUTs and the associatedcarry chains in a logic slice of modern FPGAs such as XilinxVirtex-7series.

A. AccurateSignedPartialProductsGeneration

Fig.1showstheconfigurationsofthe6-inputLUTsusedfortheimplementationoftheproposedaccurate multiplier.TheBEisimplementedbyLUTType-*A*configuration,asshowninFig.1(a).Itreceivesfiveinputs,i.e.,$a_n$anda_{n1}(frommultiplicand)andb_{m1},$b_m$andb_{m1}(frommultiplier). TheLUTinternallyimplementsthreeMUXes.Basedonthevalue ofBE,thefirstMUX(controlledbyssignal)decideswhether$a_n$or a_{n1}shouldbeforwardedforpartialproductgeneration.Thesecon dMUX,controlledby*c*signal,managestheinversionoftheoutputofthefirstMUX.Finally,thethirdMUXcanmak ethepartialproductzerodependinguponthevalueof the *z* signal. This information is forwarded to the associatedcarrychainascarryingpropagatesignal"p_{out}."Theinput$a_n$isusedasthecarrygeneratesignal"g_{out}"forthecarrychain.Bewick'sSEtechniqueforeachpartialproductrowisimplementedb yLUTType-*B*andLUTType-*C*configurations,asshowninFig.1(b)and(c),respectively.TheLUTType-*B*receivesfiveinputs,i.e.,b_{m1},b_m,andb_{m1}(frommultiplier),a_n(the MSBofthemultiplicand),andp_{in}.Thep_{in}signalisconstant"1"forth efirstrowofpartialproductsandforallotherrowsitisconstant"0."T heLUTcomputestheSEsignal,performs the XORoperation on it and provides the result to theassociatedcarrychainasthecarrypropagatesignalp_{out}.Thecarr ygenerate signal $g_{out}$is directlyprovided by the p_{in}signal.LUTType-*C*isusedtotransferthecorrectsigninformationofitsrespectiveparti alproductrowtothefollowingpartial productrow.

Utilizing LUTs of types *A*, *B*, and *C*, Fig. 2(a) shows thefirst row of partial products for an 88 multiplier. The right-most LUT of Type-*A* in each partial product row is used forcomputing the required input carry. This input carry is appliedfor representing a partial product in 2's complement format.For an 88 multiplier, a total of four partial product rows willbe generated. The last partial product row does not require anLUTofType-*C*.

B. OptimizingCriticalPathDelay

For an $N \times M$ multiplier, the length of the carry chain ineach partial product row is $N4$ bits. To improve the criticalpath delay of the multiplier, the length of the carry chain canbe reduced to $N1$ bits. A critical path delay-optimized imple-mentation of our novel multiplier is shown in Fig. 2(b). Thepartial product terms $pp_{(x,0)}$and $pp_{(x,1)}$, in each partial productrow,requireoneandtwobitsofthemultiplicand,respectively.These two partial product terms can be implemented by onesingle 6-input LUT "*A*1." Similarly, $pp_{(x,2)}$, in each partialproduct row, can be independently implemented using another6-inputLUT"*A*2."Aseparate6-inputLUT,"CG,"canbeused

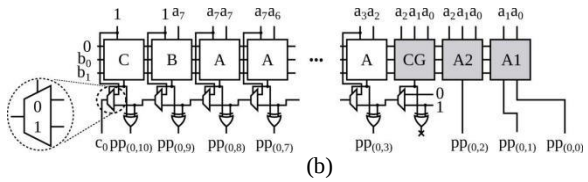

Fig.2. Firstpartialproductrowforan8 × 8multiplier.(a)Firstversionofmultiplier.(b)Optimizedversionofmultiplier.

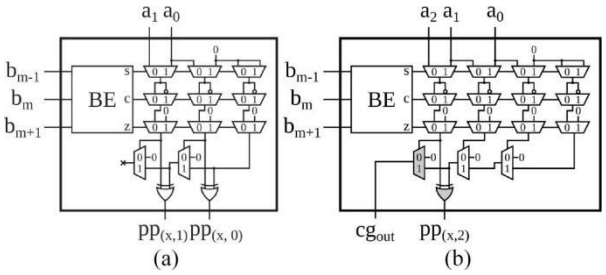

Fig.3. ConfigurationofLUTstypes*A*1,*A*2,andCG.(a)LUT*A*1.(b)LUT *A*2/CG.

tocomputethecorrectinputcarryforeachpartialproductrow.Fig. 3 shows the internal configurations of LUT types *A*1, *A*2,and CG, respectively. LUT types *A*2 and CG only differ in theoutput signals $pp_{(x,2)}$and cg_{out}. LUT type *A*2 utilizes $pp_{(x,2)}$signalsolely,whereasLUTtypeCGusescg$_{out}$signalexclusively. Foran$N \times M$multiplier, thenumberofLUTsrequiredtogeneratepartialproductsis$(N+3) \times \lceil M/2 - 1$.

C. AccumulationofGeneratedPartialProducts

For the reduction of generated partial products to computethe final product, binary adders, ternary adders, and 4:2 compressors[15]canbeutilized.A4:2compressoriscapableof reducing four partial productrows to two output rows.During our experiments, we observe that the deployment ofternary adders might reduce the overall resource utilization.However, they have higher critical path delays than binaryadders.Therefore,inthisletter,the4:2compressorsandbinaryadders are used for the reduction of the generated partial products. We have used the 6-input LUTs and the associated carrychainstoimplementthem.

III. ResultsandDiscussion

We have used VHDL for the RTL implementations of allpresented multipliers. The proposed designs have been synthesized and implemented using Xilinx Vivado 17.4 for theVirtex-7 xc7v585tffg1157-3 FPGA (unless stated otherwise).Powervaluesareestimatedbythesimulatorandpoweranalyzertoolsprovidedby Vivado.

Wehavecomparedtheimplementationresultsofourproposed multiplier with the Vivado's area/speed-optimizedmultiplierIPs[4],"R1"[8],"R5"[7],and"R7"[14].[2]Furthermore, the proposed design is also evaluated against thestate-of-the-artapproximatemultipliers"R2"[17],"R3"[2],

[2]A generic and open-source implementation for every size of multiplier isnotavailable.Signedmultiplier"mul8s_1KV8.v"fromthelibraryisused.

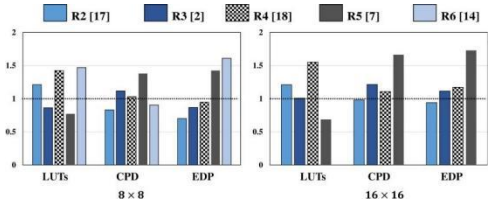

Fig.4.Comparisonoftheproposedsignedmultipliers with theunsignedmultipliers(withoutthesigned–unsignedconverters).Theresultsarenormalizedtoourproposedmultipliers.

"R4" [18], and an 88 multiplier "R6" from [14].[3] For the*unsigned* numbers-based architectures in "R1," "R2," "R3,""R4," and "R5," we have implemented signed–unsigned converters.Toshowafaircomparison,wehavereportedtheperformance results of the state-of-the-art multipliers with andwithoutusingthesesigned–unsignedconverters.

A. ImplementationResults

Table II presents the resource consumption (LUTs), CPD, andEDP requirements of our proposed design and different state-of-the-art accurate and approximate multipliers. In the table,the results for "R2," "R3," "R4," "R5," and "R6" multipliersareinclusiveofthesigned–unsignedconverters.For"R6"multiplier, there is only one design point with the input bit-widthof88.

AsshowninTableII,exceptfor"R1"[8]and"R5"[7],our proposed multiplier always requires less number of LUTsthan other state-of-the-art multipliers for different bit-widths."R1" and "R5" multipliers utilize sequential computation ofpartial products to obtain area gains at the cost of high criticalpath delays. The area savings offered by our designs increasewith the size of the multiplier, up to 16% when compared withVivado3232area/speed-optimizedIP.

Our proposed multiplier provides higher performance thanstate-of-the-art accurate and approximate multipliers. For example, compared to the 88 Vivado speed-optimized multiplierIP, our multiplier reduces the critical path delay by 21%. "R1"and "R5" accurate multipliers have higher critical path delaysamongallpresentedmultipliers.

The energy efficiency of the presented multiplier designs ischaracterized by the EDP as illustrated in Table II. It can bedrawn from the table that our proposed multipliers have betterenergy efficiency than state-of-the-art across different sizes. Forexample, our 1616 multiplier delivers up to 23.6% reductioninEDPwhenitiscomparedagainst"R1."

To further elaborate on the efficacy of our proposed implementation,TableIIshowstheaveragesoftheproductsofthe normalized values of LUTs utilization, CPD, and EDP(Average [Norm. LUTsNorm. CPDNorm. EDP]) acrossdifferent sizes of multipliers. All individual performance metricsofeachmultiplierhavebeennormalizedwithrespecttothecorresponding performancemetricsofVivadoarea-optimized multiplier IP. Our proposed multiplier outperformsstate-of-the-artimplementationsintheoverallscore.

Wehavealsocomparedourproposed*signed*multiplierstothestate-of-the-artaccurateandapproximate*unsigned*multiplierswithoutdeployingsigned–unsignedconverters.Fig. 4 presents the resource utilization, CPD, and EDP of 88and 1616proposed implementations, "R2," "R3," "R4,""R5," and "R6" multipliers. These results have been normalizedtotheimplementationresultsofourproposedimplementations.

[3]For“R6”approximate“mult_000.v”fromthelibraryisused.

TABLEII
IMPLEMENTATION RESULTS OF DIFFERENT MULTIPLIERS."R2,""R3,""R4,""R5," AND "R6" MULTIPLIERS ARE IMPLEMENTED WITH THE SIGNED–UNSIGNED CONVERTERS. THE RESULTS WITH SHADING ARE THE LOWEST IN THEIR RESPECTIVE COLUMN

Design	4x4			8x8			16x16			32x32			Average Performance
	LUTs	CPD [ns]	EDP [zJs]	LUTs	CPD [ns]	EDP [zJs]	LUTs	CPD [ns]	EDP [zJs]	LUTs	CPD [ns]	EDP [zJs]	
Ours	*18*	*1.65*	*1.86*	*66*	*2.80*	*16.97*	*243*	*4.13*	*93.61*	*928*	*6.21*	*316.43*	0.347
R1: Walters [8]	10	2.00	1.82	36	3.65	16.57	136	6.56	122.53	528	13.09	930.54	0.427
Vivado IP (Speed) [4]	18	2.14	2.27	74	3.54	20.29	286	4.27	146.62	1103	5.81	861.45	0.532
R2: Kulkarni [17]	20	2.12	2.17	86	4.89	36.31	330	6.59	134.39	1257	8.92	303.06	0.885
R3: Ullah [2]	22	3.34	4.57	81	5.19	38.48	296	7.33	136.14	1121	9.65	354.33	1.02
R4: Rehman [18]	18	2.23	2.25	92	4.99	35.43	404	7.03	156.87	1512	9.57	322.10	1.05
R5: Kumm [7]	24	3.84	9.55	73	6.08	58.95	217	9.52	313.56	700	16.25	1631.78	1.931
R6: Mrazek [14] Approx	-	-	-	121	4.21	27.18	-	-	–	-	-	-	-
R7: Mrazek [14] Accurate	-	-	-	79	3.28	34.25	-	-	–	-	-	-	-
Vivado IP (area) [4]	30	2.91	6.56	88	3.45	31.26	326	5.04	207.96	1102	6.79	1050.63	1

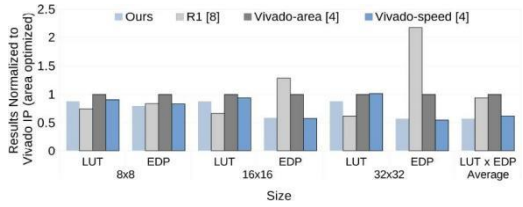

Fig. 5.Results of the neural network use-case. The LUT resources and EDPobtainedfordesignswithdifferentmultiplierarenormalizedtoVivado-area.

Compared to the accurate 16×16 "R5" multiplier, our implementation provides 39.0% and 42.0% reduction in CPD andEDP,respectively.

B. CaseStudies

1) Artificial Neural Network's Inference With Small-SizeMultiplier: We have also applied our multiplier for the inferencestageofaneuralnetwork[19].Thenetworkisusedforthecla ssificationofhandwrittendigitsfromMNISTdatabase.Theinference accuracy of the ANN× for 10 000 images with 8-b fixedpointnumbersandour88multiplieris96.67%.Theoriginalac curacy with 64-b number and multiplier is 97.09%. The lossin classification accuracy for the multiplier is negligible. If thenetwork was implemented on the FPGA with our proposed accu-rate multiplier instead of the Vivado's area-optimized multiplier,theestimatedLUTsavingis17.5%.

*2) ArtificialNeuralNetwork'sInferenceImplementationonFP GA:*The target FPGA is Xilinx Zynq Ultrascale xczu3egused in the Ultra96 evaluation platform. The network has onefully connected layer. Inside each neuron, beside the MACunit, there are also activation and quantization modules. TheactivationfunctionisReLU.Thequantizationmoduleconverts the MAC results (which are represented in a wider bit width)backtotheoriginalfixed-pointformat.

First, we implement the network with the Vivado's speed-optimizedmultiplierwithasmanynumberofneuronsaspossibleint hreedifferentinputsizes,8 8,16 16,and × × 3232. The timing constraint is 4 ns. After that, the samesetups are applied for the Vivado-area multiplier, R6 [9], andours. The results are presented in Fig. 5. In the combinedLUTxEDP average across all input sizes, ours offers the bestresults.Ourmultiplieris5%,43%,and37%betterthanVivado-speed,Vivado-area,andR6[9],respectively.WhileR6[9]hasthe lowest LUT counts, its EDP is the worst among all whenthe input size increases. In comparison with Vivado-speed,oursiscomparableinEDPbutrequiresanaverageof8%less number of LUTs. These results imply that with the sameamount of fixed FPGA resources, more of our multipliers canbe instantiated to further exploit the available parallelism oftheapplicationwithonlyaslightincreaseinenergy(ifany).

Ourmultiplieralsofitswellwithvariousmodern XilinxFPGAarchitectur es.

IV. CONCLUSION

A unique area-optimized, low-latency, and energy-efficient accurate signed multiplier architecture for FPGA-based systems was given in this letter. The advantages of our multipliers in neural network applications have also been assessed. We will make available the RTL models of our designs at https://cfaed.tu-dresden.de/pd-downloads as an open-source library..

REFERENCES

[1] *7SeriesDSP48E1Slice*,documentUG479,Xilinx,SanJose,CA,USA,2018.
[2] S. Ullah *et al.*, "Area-optimized low-latency approximate multipliers forFPGA-based hardware accelerators," in *Proc. ACM/ESDA/IEEE DesignAutom.Conf.(DAC)*,2018,pp.1–6.
[3] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs,"*IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2,pp.203–215,Feb.2007.
[4] *LogiCOREIPMultiplierv12.0*,documentPG108,Xilinx,SanJose,CA,USA, 2015.
[5] A.D.Booth,"Asignedbinarymultiplicationtechnique,"*Quart.J.Mech.Appl. Math.*,vol.4,no.2,pp.236–240,1951.
[6] C. R. Baugh and B. A. Wooley, "A two's complement parallel arraymultiplicationalgorithm,"*IEEETrans.Comput.*,vol. C-22,no. 12,pp.1045–1047,Dec.1973.
[7] M.Kumm,S.Abbas,andP.Zipf,"Anefficientsoftcoremultiplierarchitecture forXilinxFPGAs,"in*Proc.IEEESymp.Comput.Arithmetic(ARITH)*,2015,p p.18–25.
[8] E. G. Walters, "Array multipliers for high throughput in Xilinx FPGAswith6-inputLUTs,"*Computers*,vol.5,no.4,p.20,2016.
[9] H.Parandeh-AfsharandP.Ienne,"Measuringandreducingtheperformance gap between embedded and soft multipliers on FPGAs," in*Proc.Int.Conf.FieldProgram.LogicAppl.(FPL)*,2011,pp.225–231.
[10] *7 Series FPGAs Configurable Logic Block*, document UG474, Xilinx,SanJose,CA,USA,2016.
[11] H.Parandeh-Afshar,P.Brisk,andP.Ienne,"Exploitingfastcarry-chainsof FPGAs for designing compressor trees," in *Proc. Int. Conf. FieldProgram.LogicAppl.(FPL)*,2009,pp.242–249.
[12] A. Kakacak, "Fast multiplier generator for FPGAs with LUT basedpartial product generation and column/row compression," *Integr. VLSIJ.*,vol.57,pp.147–157,Mar.2017.
[13] M.Kumm,J.Kappauf,M.Istoan,andP.Zipf,"Resourceoptimaldesign of large multipliers for FPGAs," in *Proc. IEEE Symp. Comput.Arithmetic(ARITH)*,2017,pp.131–138.
[14] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b:Library of approximate adders and multipliers for circuit design andbenchmarking of approximation methods," in *Proc. Design Autom. TestEuropeConf.Exhibit.(DATE)*,2017,pp.258–261.
[15] M.KummandP.Zipf,"EfficienthighspeedcompressiontreesonXilinxFPG As,"in*Proc.MethodsDescriptionLang.Model.VerificationCircuitsSyst.(M BMV)*,2014,pp.171–182.
[16] G. W. Bewick, "Fast multiplication: Algorithms and implementation,"Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA,USA,1994.
[17] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for powerwith an underdesigned multiplier architecture," in *Proc. 24th Int. Conf.VLSIDesign*,2011,pp.346–351.
[18] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, and J. Henkel,"Architectural-space exploration of approximate multipliers," in *Proc.EEE/ACMInt.Conf.Comput.-AidedDesign(ICCAD)*,2016,p.80.
[19] *MNIST-cnn*. (2016). [Online]. Available: https://github.com/integeruser/MNIST-cnn