Implementation and Physical Design of 8/4-Bit Signed Divider

Shubham Purohit
VLSI & Embedded Systems Group
DA-IICT
Gandhinagar, India
201911041@daiict.ac.in

Prashant Laddha
VLSI & Embedded Systems Group
DA-IICT
Gandhinagar, India
201911023@daiict.ac.in

Rutu Parekh
VLSI & Embedded Systems Group
DA-IICT
Gandhinagar, India
rutu_parekh@daiict.ac.in

2021 8th International Conference on Signal Processing and Integrated Networks (SPIN) | 978-1-6654-3564-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/SPIN52536.2021.9566020

***Abstract*—This paper focuses on implementing a signed binary divider using Verilog and performing a physical design process i.e. register transfer level (RTL) to graphic design system-II (GDSII) on 180nm and 45nm technology nodes to analyze different parameters such as delay, area, power, and bandwidth. It also extends the unsigned divider to the signed divider and hence increases the range of division. On 180nm, this divider consumes 14263.2 μm^2 area, the power consumption of 20 μW and bandwidth of 666.66 Mhz, whereas for 45nm node area consumption of this divider is 600.130 μm^2 with a power consumption of 17.88 μW and bandwidth of 892Mhz.**

***Index Terms*—Non-Restoring, Signed divider, Physical design**

I. Introduction

In recent years, the arithmetic logical unit (ALU) has been changed many times. A significant change is observed in the implementation of an individual multiplication and division unit of the ALU. Algorithms for division units also get updated from time to time for different applications. In a typical program, an arithmetic unit produces thousands of division computations per second. To compute and produce correct results, the ALU's division algorithms must be as effective as possible. Since the age of computer arithmetic, researchers have designed many algorithms to divide numbers. Some algorithms work well for hand analysis, and some are good for hardware computation.

Different types of division algorithms for large division are explained in [1], namely radix-2 restoring division, radix-4 restoring division, radix-4 non-restoring division. The restoring division algorithm is more complicated than the non-restoring algorithm but gives better results in division. The 16-bit division is implemented on Verilog using Xilinx ise in [2]. Due to the simplicity of the non-restoring division algorithm, as it only requires adders and subtractors and does not require multipliers, it is preferred for unsigned division [3]. Radix-4 and radix-8 parallel dividers are implemented in [3]. These high radix dividers are also experimentally fabricated using 0.35µm CMOS technology [3]. Reddy et al., focus on implementing an approximate divider with both non-restoring and restoring algorithms. This approximate divider consumes only 31% and 33% of delay and power, respectively as compared to conventional dividers, at the cost of reduced efficiency [4]. The novel radix-4 division is implemented in [5] using recurrence architecture, which contains signed-digit adders. Power and area for the same divider depicted in [5] are calculated on a 45nm technology node from global foundries.

Reza et al. propose a new algorithm, namely speed yet energy-efficient rounding-based approximate divider(SEERAD), useful for digital signal processing(DSP) applications [6]. Vedic divider using parvartya sutra is implemented in [7], [8]. Vemula et al. focus on the physical design of 64-bit Vedic divider using the Cadence Encounter tool. Difficulties related to physical design for a large circuit with billions of transistor is described in [9].

A high-speed, scalable programmable divide-by-N frequency divider is presented in [10]. This divider includes a new proposed state look-ahead parallel counter with a basic conventional D-type Flip-Flop (DFF) circuit [10]. The divider developed in [10] using 150nm TSMC library can operate up to 2 GHz for a 1.35V supply voltage with a maximum power consumption of 16.78mW and a measured area of 13585μm^2 [10]. Non restoring and restoring array dividers has been implemented using cmos and cpl based adder cell and subtractor cell respectively [11]. These dividers are implemented in Verilog and layout is also constructed using 65nm technology in [11]. Design of full-adder using by Shannon theorem based on the pass transistor approach and eventually designs the non-restoring divider using the adder developed in [12]. Parameters of divider presented in [12] such as power, area, delay, latency, throughput are also calculated on 180nm technology node. Venkatachalam et al. proposed two approximation models for restoring dividers [13].

To get the fabricated chip of any VLSI design module there are multiple steps, the first step is the RTL coding and synthesis followed by physical design steps in which the last step is to get a graphic design system-II (GDS-II) file. There are many tools available from Cadence, Synopsis, Mentor graphics for the physical design, but these tools are very much costly so as an alternative some free open-source EDA tools are also emerging nowadays. This paper focuses on extending the above-unsigned division functionality of [2] to a signed division and analyzes the performance of a signed division with open-source EDA tools. Open-source EDA tools such as qflow are explained in [14]- [15], from where information about physical design can be gathered.

TABLE I
SIGN OF QUOTIENT AND REMINDER FOR DIFFERENT COMBINATION OF DIVIDEND AND DIVISORS

Dividend	Divisor	Quotient	Remainder
Positive	Positive	Positive	Positive
Positive	Negative	Negative	Positive
Negative	Positive	Negative	Negative
Negative	Negative	Positive	Negative

This paper is organized as follows: The proposed algorithm is discussed in section II. Next, section III contains stepwise information on physical design flow. Different design parameters with varying technology nodes are discussed in section IV. Conclusions are drawn in section V. Finally, section VI contains References referred to for this work.

II. PROPOSED ALGORITHM FOR SIGNED DIVIDER

The proposed architecture for the signed binary divider uses the non-restoring divider logic as a base block. In this work the size of the dividend is 8-bit and the size of the divisor is 4-bit. The algorithm is demonstrated in Fig. 1. The steps of a block diagram are explained below. In Fig. 1 D_{n-1} to D_0 are bits of Dividend$_p$ in this paper it is given as D_7 to D_0 as we designed the divider for an 8-bit dividend.

1) First based on sign bit of dividend and divisor, positive values of dividend and divisor is processed.
2) These positive values are then going to non-restoring divider logic.
3) Quotient and reminder generated from divider logic are still incorrect.
4) Quotient and the remainder is calculated with the help of sign bits of dividend and divisor. The Table I for the same is mentioned below.

As an example take a divisor as $(5)_{10}$ or $(0101)_2$ and dividend as $(26)_{10}$ or $(0001\_1010)_2$. As described in the algorithm. (Note: initial value of quotient is zero).

1) First, the value of $dividened_(p)$ and $divisor_(p)$ is selected based on dividend and divisor sign bit. In this example, both the values are positive, hence there will be no change in the dividend and divisor's value. Moreover, the sign bit is stored as 0 for both.
2) Now, subtract the top four MSB from the divisor. Which is 0001 - 0101.
3) The subtraction result is negative. Now the quotient is left-shifted by 1, so there will be no change in the quotient. The count is decreased by 1.
4) Now the dividend is left-shifted by 1, therefore the new dividend is 0011_0100.
5) The same procedure is repeated. As the top 4 MSB bits of dividend (0011) is less than the divisor, we can shift the quotient, and dividend by 1.count is also decreased by 1.
6) The new dividend is 0110_1000. Now the subtraction of dividend top 4 MSB and divisor is positive. So the quotient will be updated to 0001.
7) Now, the top 4 MSB of dividend is replaced by subtraction result, which is 0001. Hence the updated dividend is 0001_1000.count is also decreased by 1, and the count is not zero, so left shift the dividend.
8) The new dividend is 0011_0000. As the top 4 MSB bits of dividend (0011) is less than the divisor, therefore, we can left shift the quotient and dividend by 1.count is also decreased by one, and dividend is left-shifted by 1.
9) The new dividend is 0110_0000. Now the subtraction of dividend top 4 MSB and divisor is positive. So the quotient will be updated to 0101.
10) Now, the top 4 MSB of dividend is replaced by subtraction result, which is 0001. Hence the updated dividend is 0001_0000.count is also decreased by 1, and the count is zero now.
11) now results selected based on the value of sign bits of Dividend and divisor. As both Dividend and divisor are positive, the resultant quotient is 0101, and the remainder is 0001.

III. PHYSICAL DESIGN FLOW

Physical design is converting an RTL gate-level netlist into planar geometric shapes GDSII file, which can be used by foundries for fabrication. It contains different steps, like floorplanning, placement, clock tree synthesis, and routing. All steps of physical design are shown in Fig.2. Physical design starts with a netlist, the netlist is synthesized from RTL, and it describes the circuit components in a manner how they are connected.

Step 1: Synthesis: Synthesis is a process of converting the Verilog code into a logic gate-level netlist. The synthesis tool requires two input files. First is the technology library file(.lib), which contains standard cells. The second file is a constraint file(.sdc), including timing, loading, and optimization algorithm for logic optimization. After synthesis, the next step is floor Planning.

Step 2: Placement: The location of every component on the die can be determined by Placement by considering the timing data and length of interconnects. To optimize the placement step, it is divided into 4 phases: 1. Pre-placement optimization, 2. In placement optimization 3. Post-placement optimization (PPO) before clock tree synthesis (CTS) 4. Post-placement optimization after CTS.

Step 3: Clock tree synthesis (CTS): CTS's main objective is to minimize the skew and latency by inserting inverters or buffers so that the clock shall be distributed equally.

Step 4: Routing: Routing aims to decide the interconnects paths, which includes macro pins & standard cells. Routing makes all the connections described in the netlist too effectively without violating setup and holding time constraints.

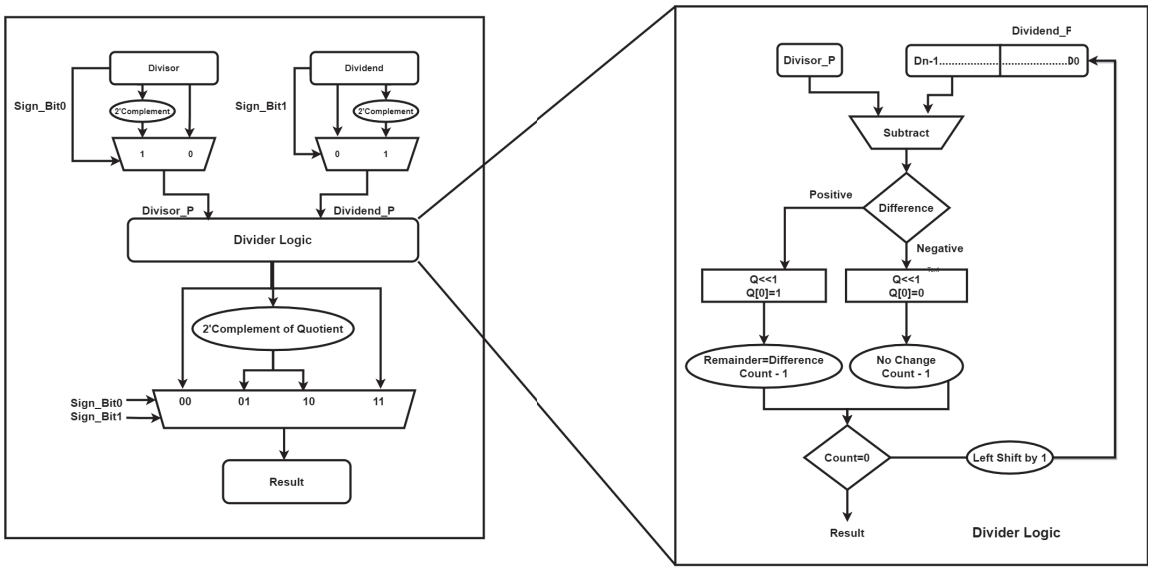

Fig. 1. Proposed design Algorithm of divider.

Step 5: Physical Verification: Physical verification checks whether the generated design is correct or not.

RTL of our proposed design is converted to GDSII by passing through various steps using the free EDA tool i.e., Qflow. This tool takes RTL code as an input, which is synthesized in yosys, and generates the synthesized netlist. This netlist is then given as an input to the next tool in the process and at last, the GDSII layout is attained [11].

Tools [11] used in the RTL-to-GDSII conversion of the design are:

1) Yosys: For RTL synthesis

2) Graywolf: For placement

3) Qrouter: For routing

4) Magic: For layout editing

5) OpenSTA : For static timing analysis(STA)

IV. Results of RTL to GDDS-II Steps for Developed Divider

The whole RTL to GDS-II flow comprises different steps divided into two main sections: front end design and back end design, which will be done by various open tools as described below.

A. Front end design using Open Source EDA tools

1) RTL simulation: To check the functionality of the circuit proposed in this paper xilinx ise tool was used which gives the output waveform for certain inputs as shown in below Fig. 3

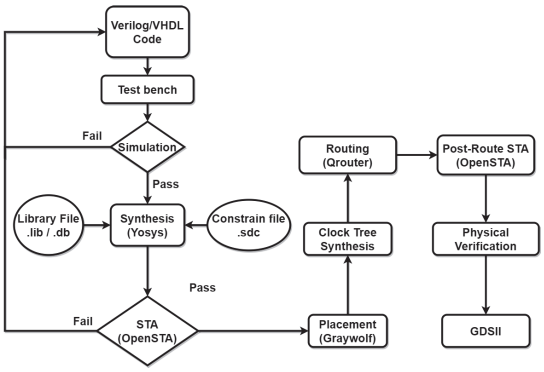

Fig. 2. Physical design flow.

2) Synthesis: Open source tool yosys is used for generating synthesizable verilog code which is necessary for timing analysis in the back end flow.
Several standard cells in the proposed design can be seen from Fig. 4.

3) Static timing analysis: Static timing analysis(STA) is required to check the setup and hold time constraints of the design.
STA is performed by OpenSTA, a free, open-source tool. it is seen from Fig. 5 that for minimum delay path data arrival time is more than data required time and also the slack time is positive which stats that hold time constraint is met.

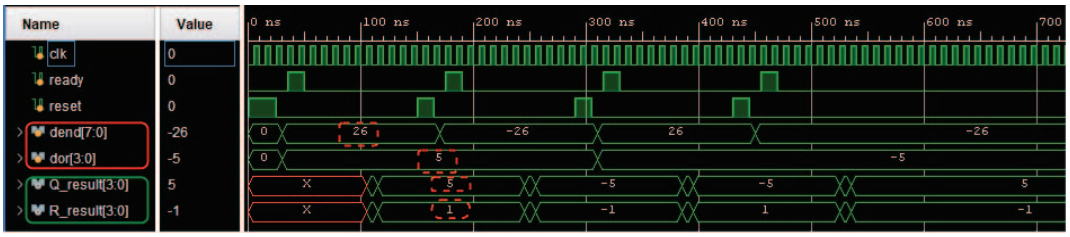

Fig. 3. RTL Simulation.

```
      AND2X2 cells:        4
     AOI21X1 cells:        9
     AOI22X1 cells:        2
       INVX1 cells:       30
      MUX2X1 cells:        1
     NAND2X1 cells:       26
     NAND3X1 cells:        6
      NOR2X1 cells:       16
      NOR3X1 cells:        1
     OAI21X1 cells:       46
     OAI22X1 cells:        7
       OR2X2 cells:        3
     XNOR2X1 cells:        7
      XOR2X1 cells:        7
 internal signals:      181
    input signals:       54
   output signals:       33
```

Fig. 4. Number of standard cells required in the design.

```
Path Type: max

  Delay    Time   Description
---------------------------------------------------------
   0.00    0.00   clock clk (rise edge)
   0.00    0.00   clock network delay (ideal)
   2.00    2.00 ^ input external delay
   0.00    2.00 ^ ready (in)
   0.18    2.18 v _196_/Y (INVX1)
   0.45    2.63 ^ _207_/Y (OAI21X1)
   0.26    2.89 v _208_/Y (INVX1)
   0.19    3.07 ^ _264_/Y (NAND3X1)
   0.05    3.12 v _296_/Y (OAI21X1)
   0.00    3.12 v _330_/D (DFFPOSX1)
           3.12   data arrival time

  10.00   10.00   clock clk (rise edge)
   0.00   10.00   clock network delay (ideal)
   0.00   10.00   clock reconvergence pessimism
          10.00 ^ _330_/CLK (DFFPOSX1)
  -0.16    9.84   library setup time
           9.84   data required time
---------------------------------------------------------
           9.84   data required time
          -3.12   data arrival time
---------------------------------------------------------
           6.72   slack (MET)
```

Fig. 6. Slack time for maximum delay path.

it is seen from Fig. 6 that for maximum delay path data arrival time is less than data required time and also the slack time is positive which stats that setup time constraint is met.

B. Back End Design using Open Source EDA tools

1) Placement: Location of every component on the die can be determined by Placement by considering the timing data and length of interconnects. To optimize the placement step, it is divided into 4 phases:
Pre-placement optimization, in placement optimization, post-placement optimization before CTS, post-placement optimization after CTS.
We carried out the placement step using the qflow tool with an aspect ratio setting of 0.5, 1, 0.75 and an initial density setting of 1. Initial density and aspect ratio do not make much difference in the physical design layout area's size.

```
Path Type: min

  Delay    Time   Description
---------------------------------------------------------
   0.00    0.00   clock clk (rise edge)
   0.00    0.00   clock network delay (ideal)
   0.00    0.00 ^ _314_/CLK (DFFPOSX1)
   0.10    0.10 ^ _314_/Q (DFFPOSX1)
   0.00    0.10 ^ _351_/D (DFFSR)
           0.10   data arrival time

   0.00    0.00   clock clk (rise edge)
   0.00    0.00   clock network delay (ideal)
   0.00    0.00   clock reconvergence pessimism
           0.00 ^ _351_/CLK (DFFSR)
   0.01    0.01   library hold time
           0.01   data required time
---------------------------------------------------------
           0.01   data required time
          -0.10   data arrival time
---------------------------------------------------------
           0.10   slack (MET)
```

Fig. 5. Slack time for minimum delay path.

2) Routing: The goal of routing is to decide the interconnect paths, including macro pins and standard cells. Routing makes all the connections described in the netlist too effectively without violating setup and holding time constraints.

3) Migration: Migration is used to convert DEF files produced by routing into a database for the magic layout editor. In this step, a layout can be corrected manually to avoid errors in design rule check(DRC) and layout vs. schematic(LVS). Fig. 7 shows the layout of developed signed divider on 180nm technology node with an aspect ratio of 1 and initial density is also 1. The layout contains standard cells i.e. NAND, NOR, AOI, OAI.

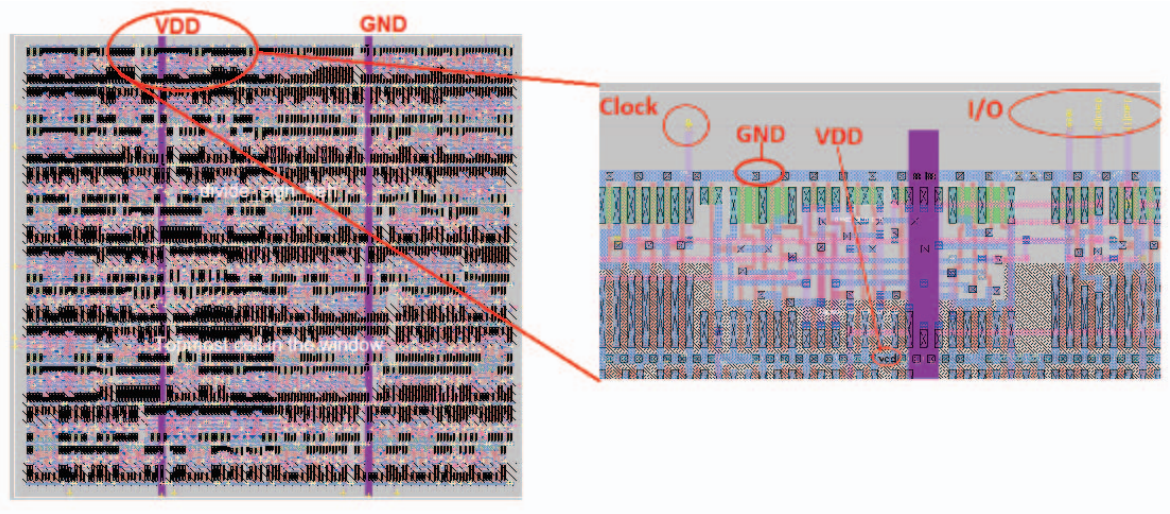

Fig. 7. Layout of proposed design.

TABLE II
PARAMETERS ON DIFFERENT TECHNOLOGY NODES

Reference	Technology node	Area (μm^2)	f_{max} (Mhz)	delay (ps)	Power at f_{max} (μW)
This work	180 nm	14263.20	666.66	8.463	20
This work	45 nm	600.130	892	6.2	17.88
Ref [4]	45nm	–	–	1256	144.5
Ref [5]	45 nm	1131.883	–	–	708.00
Ref [11] (non-restoring)	65 nm	–	–	417	370.42
Ref [11] (restoring)	65 nm	–	–	450	164
Ref [12]	180 nm	46552	1330	250	64.08

C. Comparison Table

Various design corners of the proposed divider system are computed and listed, as shown in Table II. Where the clock period for the divider is kept as 10ns. It is seen from Table II that design on a 180nm technology node consumes 14263.20 μm^2 area and consumes 20 μW power at a maximum frequency of 666.66 Mhz. While the same design on a 45nm technology node consumes only 600.130 μm^2 area, 17.88 μW power at a maximum frequency of 892 Mhz. From the comparision of results using both techonolgy nodes it can be stated that though power consumption is not reduced too much but the significant amount of area can be saved if this design is fabricated using 45nm technology node. Note that the divider's physical design on the 45nm technology node is performed using cadence tools. In addition to that, it can also be seen from Table II that the power consumed by the divider developed in this work is lesser than the power consumption of circuits that are referred for this work. Though, the power in this work can be lesser due to lesser dividend and divisor bits. It can be seen from Table II that the design proposed in this work operates on a relatively low frequency than the design proposed in the [12] even for the same technology node i.e. 180nm. It is due to the fact that physical design steps of divider is not performed. So, the design does not includes the effect of interconnects and thus the RC parasitics. On the other hand design proposed in this work considers interconnects and due to that the operating frequency is low as shown in this paper. Also, the design proposed in this work consumes lesser power than the design in [12]. Because, as the operating frequency increases the power will increase in exponential manner. So it is a trade off between power and frequency.

V. CONCLUSION

We have successfully implemented RTL to GDSII physical design flow for the proposed signed binary divider. Different design parameters such as area, power, delay, bandwidth on different technology nodes i.e., 180nm, 45nm. This work has been Performed using the qflow tool, which comprises various open-source tools. At 45nm proposed design is 10.6% power-efficient, 95% area-efficient, and 26.74% timing efficient compared to design at 180nm node. Also, the proposed algorithm consumes low power and area. So, it can be used for very low-power integer division applications.

REFERENCES

[1] H. Kaur, "Performance Analysis Of Various Multiplication And Division Algorithms For Large Numbers"

[2] Y.Yusmardiah, et al., "Translation of Division Algorithm Into Verilog HDL," *ARPN Journal of Engineering and Applied Sciences*, vol. 12, pp. 3214–3217, 2006.
[3] T. Aoki, K. Nakazawa, and T. Higuchi, "High-radix parallel VLSI dividers without using quotient digit selection tables," *in Proc. 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000)*, pp.345-352, 2000.
[4] K. Reddy, et al., "Design of approximate dividers for error-tolerant applications," *in Proc. 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 496-499, 2018.
[5] Gaalswyk, F. Matthew, and J. Stine, "A Low-Power Recurrence-Based Radix 4 Divider Using Signed-Digit Addition,"*in proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 391-396, 2019.
[6] R. Zendegani, et al., "SEERAD: A high speed yet energy-efficient rounding-based approximate divider," *in proc. Design, Automation & Test in Europe Conference & Exhibition*, pp. 1481-1484, 2016.
[7] R. Vemula, et. al., "Design and Implementation of 64 Bit Divider Using 45nm CMOS Technology," *International Journal of Pure and Applied Mathematics*, 2018.
[8] S. Panda, and A. Sahu, "A novel Vedic divider architecture with reduced delay for VLSI applications," *International Journal of Computer Applications*, 2015.
[9] P. Groeneveld, "Physical design challenges for billion transistor chips," *in proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors. IEEE*, pp. 78-83, 2002.
[10] A. Saleh, et al., "High speed digital CMOS divide-by-N frequency divider," *IEEE International Symposium on Circuits and Systems*, pp. 592-595, 2008.
[11] M. Basha, et al., "Novel Low Power and High speed array divider in 65 nm Technology," *International Journal of Advances in Science and Technology*, pp.44-56.
[12] C. Senthilpari, et al., "Lower delay and area efficient non-restoring array divider by using Shannon based adder technique," *in proc. IEEE International Conference on Semiconductor Electronics*, pp. 140-144, 2010.
[13] S. Venkatachalam, et al., "Design of approximate restoring dividers," *IEEE International Symposium on Circuits and Systems*, pp. 1-5, 2019.
[14] J. Jung, et al., "OpenDesign flow database: the infrastructure for VLSI design and design automation research," *in proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2016.
[15] K. Ghosh, and A. Ghosh, "Technology mediated tutorial on RISC-V CPU core implementation and sign-off using revolutionary EDA management system (EMS)—VSDFLOW," *in proc. China Semiconductor Technology International Conference (CSTIC)*, IEEE, pp.1-3, 2018.
[16] J. Dean, D. Patterson, and C. Young, "A new golden age in computer architecture: Empowering the machine-learning revolution," *IEEE Micro* , vol. 38, no. 2, pp. 21-29, 2018.
[17] W. Lee, "Verilog coding for logic synthesis," *Wiley-Interscience*, 2003.
[18] J. Hennessy, and D. Patterson, "Computer architecture: a quantitative approach," *Elsevier*, 2011.
[19] G. Khosrow, "Physical design essentials," *Springer Science+ Business Media, LLC*, 2007.
[20] A. Kahng, et al., "VLSI physical design: from graph partitioning to timing closure," *Springer Science & Business Media*, 2011.