

Secure Hash Algorithm SHA-256 Based Hardware Acceleration Using Spartan-6

Hitesh M A, Mahendra R, Sumanth H S, Subodh Kumar Panda

Cite as: Hitesh M A, Mahendra R, Sumanth H S, & Subodh Kumar Panda. (2023). Secure Hash Algorithm SHA-256 Based Hardware Acceleration Using Spartan-6. International Journal of Microsystems and IoT, 1(3), 121-126. <https://doi.org/10.5281/zenodo.8315154>




© 2023 The Author(s). Published by Indian Society for VLSI Education, Ranchi, India



Published online: 21 August 2023.



Submit your article to this journal: 



Article views: 

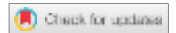


View related articles: 



View Crossmark data: 

DOI: <https://doi.org/10.5281/zenodo.8315154>



Secure Hash Algorithm SHA-256 Based Hardware Acceleration Using Spartan-6

Hitesh M A, Mahendra R, Sumanth H S, Subodh Kumar Panda

B N M Institute of Technology Bengaluru, India

ABSTRACT

Recently, there have been many technological advancements in communication, particularly in online transactions, increasing the necessity for highly secure networks and transactions. The number of cryptographic algorithms has expanded. Cryptographic hash functions are used to secure and authenticate data and transactions. SHA-256 (Secure Hash Algorithm-256) is a one-way hash function that is both secure and quick, with a high collision resistance. In this work SHA-256 hardware architecture with minimal power consumption and area based on a sequential calculation of the message scheduler and working variables. The hardware was designed in HDL and built on a SPARTAN-6 FPGA, which provides exceptional efficiency and performance. Different optimization techniques, such as gated clock conversion, arithmetic resource sharing, and structural modelling of small building blocks, were employed to further reduce power and area. The proposed design ran with a maximum frequency of 83.33 MHz the implementation reports indicated a dynamic power consumption of 14 mW and area utilization of 505 slices while maintaining a good throughput of 2659.42 G bits/s and a relatively high efficiency of 5.266 M bits/s per slice.

KEYWORDS

Acceleration; Cryptographic; FPGA; Hash; Secure; Throughput.

1. INTRODUCTION

Due to the rapid development and huge growth of the digital world, the advancement of technology and faster internet, information security has become a necessity. The idea behind our project is to speed up the hardware in such a way that has a high collision resistance and being both incredibly secure and quick. We would also like to optimize the hardware in a way that consumes less power and area which can pave paths for different applications and opportunities. This algorithm is widely used in different applications, banking, shopping, and different money related activities [1-4]. It is also used recently in blockchain technologies and internet of things applications. Information security has become essential because of the digital world's quick expansion and enormous growth as well as the increase in internet speed [5]. Our project's goal is to accelerate the hardware in a way that makes it highly secure, quick, and collision resistant. Additionally, we want to optimize the hardware so that it uses less space and power [6], opening doors for new uses and opportunities. This method is commonly utilized in many applications, including banking, shopping, and other activities involving money. Recently, it has also been employed in applications for block chain technology and the internet of things. SHA is a cryptographic hash function [7] that is widely used in various applications to provide secure

authentication, integrity, and confidentiality of digital data. Some common applications of SHA include Digital Signature Verification, Password Hashing, SSL Handshake, Integrity Checks and Blockchain Technology.

To take advantage of the hardware acceleration, an FPGA (field-programmable gate array) will be employed in the implementation. It is suitable for implementing cryptographic algorithms and performing jobs more efficiently, allowing design optimization. FPGA encryption is roughly 20 times faster than dual-core processor encryption while utilizing 85% less CPU [8-13]. Furthermore, it gives a quicker design time, greater flexibility, and reduced expenses.

2. SHA-256 ALGORITHM

2.1. Padding

Adding bits to our original message to make it the same length as the standard length needed for the hash function is the first phase of our hashing algorithm. To do this, we add a few details to the message that we already have. We compute the number of bits to add so that, after addition, the message's length should be exactly 64 bits shorter than a multiple of 512 [14-18].

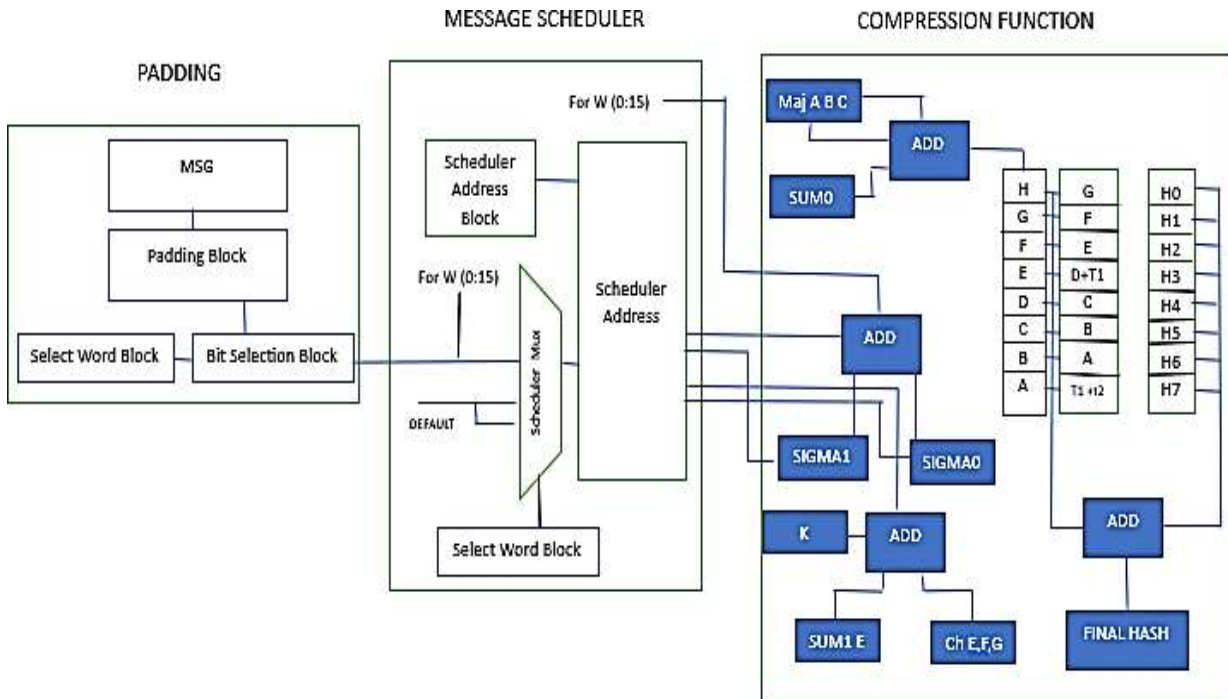


Fig.1 Block Diagram of SHA-256

2.2. Message Scheduler

A 32-bit x 16-bit (512-bit) data block serves as the message input for rounds 1 through 16. It is necessary to derive $W[i]$ for rounds 17 through 64. It is possible to determine $W[i] + K[i]$ for each round before the round iteration itself since the constant K is defined for each round.

Before the previous round has concluded, the subsequent one cannot begin. It will be advisable to keep the number of cycles each round at $1 + 3$ if your single instruction multiple data (SIMD) units have pipelines with 1 and 3 cycles [19]. The prior design utilized a set of instructions on a 128-bit register [20] set with two operands that, with careful state variable partitioning, the injection of WK values, and multiply rounds per instruction. And lacks efficient schemes for performing these operations.

2.3. Compression Function

The hash computation proceeds as follows:

For $i = 1$ to N

{

The working variables, registers A, B, \dots, H are initialized with the $(i - 1)$ st intermediate hash value (the initial hash value when $i = 1$). Apply the SHA-256 compression function to update A, B, \dots, H

For $j = 0$ to 63

{

Compute $W_j, Ch(E, F, G), Maj(A, B, C), \sum_0(A)$,
and $\sum_1(E)$

$$T1 = Kj + Wj + H + \sum_1(E) + Ch(E, F, G)$$

$$T2 = Maj(A, B, C) + \sum_0(A)$$

$$H = G$$

$$G = F$$

$$F = E$$

$$E = D + T1$$

$$D = C$$

$$C = B$$

$$B = A$$

$$A = T1 + T2$$

}

Compute the i th intermediate hash value (i) as the sum of the previous hash and the registers A, B, \dots, H . After the N iterations, the hash of the message is $(N) = (H(N), H(N), \dots, H(N))$.

Where the definitions of the logical functions are:

$$\sigma_0(x) = S_{18}(x) \wedge S_{7}(x) \wedge R_{10}(x)$$

$$\sigma_1(x) = S_{19}(x) \wedge S_{17}(x) \wedge R_{10}(x)$$

$$\Sigma_0(x) = S_{22}(x) \wedge S_{13}(x) \wedge S_2(x)$$

$$\Sigma_1(x) = S_{25}(x) \wedge S_{11}(x) \wedge S_6(x)$$

$$Ch(x, y, z) = (\neg x \& z) \vee (x \& y)$$

$$Maj(x, y, z) = (x \& y) \vee (y \& z) \vee (x \& z)$$

Rn is right shift by *n* bits

Sn is right rotation by *n* bits

The sequence of constant words K_0, \dots, K_{63} , are the first thirty-two bits of the fractional parts of the cube roots of the first sixty-four primes.

3. IMPLEMENTATION

3.1. Padding

In our design, in the padding block, on the rising edge of the first clock, the input message is padded and parsed into a 512-bit block, it then goes through the bit selection block to be sent to the message scheduler during the next 16 clock cycles in the form of 32-bit words [21].

3.2. Message Scheduler

The main role of the message scheduler block is to determine W_t to compute the working variables and the intermediate hash values.

In normal designs, the 64 message words are computed, then are used to determine the 64 working variables. On the contrary, in our proposed design, only one message word is computed and used to determine its corresponding working variables, all in one cycle. This in turn had a huge effect on throughput, area, and power. Instead of computing all working variables in 128 clock cycles, 64 for message words, and 64 for the actual working variables, the whole operation now only takes 70 cycles, nearly increasing the throughput. It also decreased the area used in the W_t register and its switching activity, resulting in huge power reduction.

3.3. Compression Function

The compression block contains combinatorial functions used to calculate the variables, and a register memory storing the values of K constants, which reduces the access time to the K_{const} .

The block receives one W_t per cycle, along with values from K_{const} , Ch , Maj , sum_1 and sum_2 , then it computes the 32-bit temporary variables which in turn update the working variables at the rising edge of each clock [21-25]. At the end of the 64th clock cycle, the working variables are added to the intermediate hash values to output the final hash [26-32].

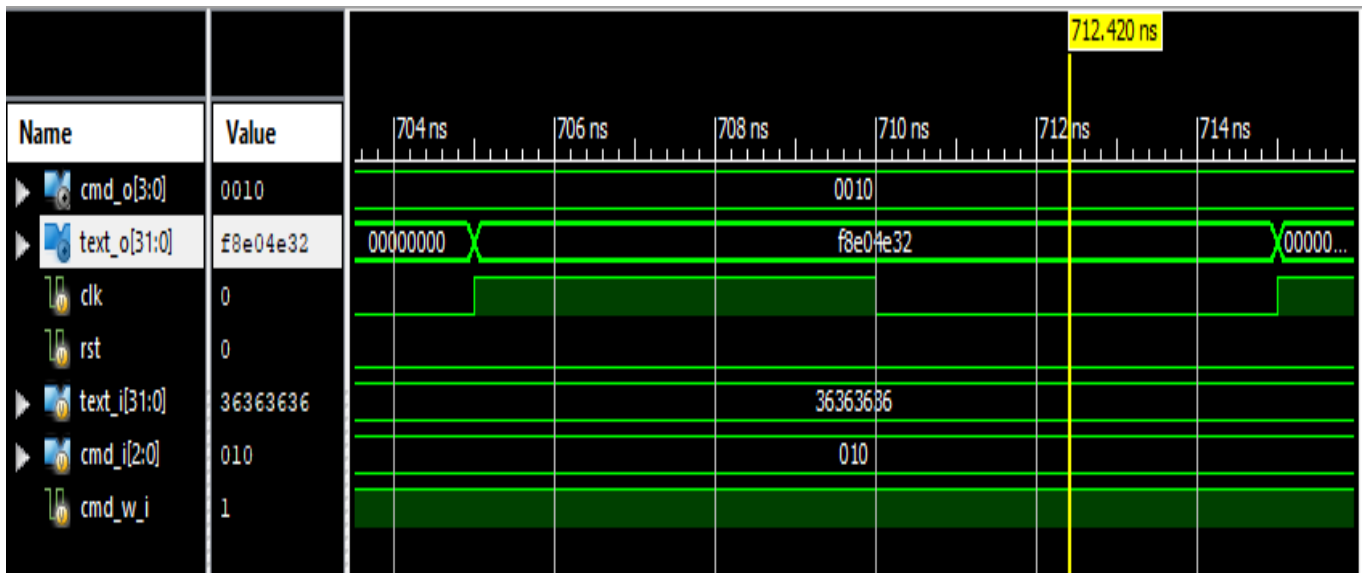


Fig.2 Post Implementation Timing Results

4. RESULTS AND DISCUSSIONS

The SHA-256 design was described using Verilog. The design was synthesized and implemented on Spartan-6 FPGA (xc6slx4-2tqg144-2), using Xilinx ISE Design Suite 14.7. The SHA-256 architecture processes a 512-bit block within 70 clock cycles. The message padding takes 1 cycle, hash computation takes 64 cycles, and outputting the final hash

takes 2 cycles. The implemented hardware achieved a maximum frequency of 83.107 MHz with a dynamic power of 14 mW using 505 slices (in comparison with 1373 slices there is a 64% reduction), 1397 LUTs, and 931 ffs. A throughput of 2659.4 Mbps and an efficiency of 5.266 Mbps/slice per slice is achieved.

Table. 1 Comparison of Results

Parameter	Our work	[1]	[2]	[3]	[4]	[5]
Slices	505	755	1373	-	-	610
LUTs	1397	-	-	2207	2150	-
Flip Flops	931	-	-	-	-	-
Maximum frequency (MHz)	83.107	174	133	74	143.164	70.55
Throughput (G bits/s)	2659.42	1370	1009	291	909.816	1344.98
Efficiency (M bits/s)	5.266	-	-	-	-	-
Power (mW)	14	-	-	-	-	-

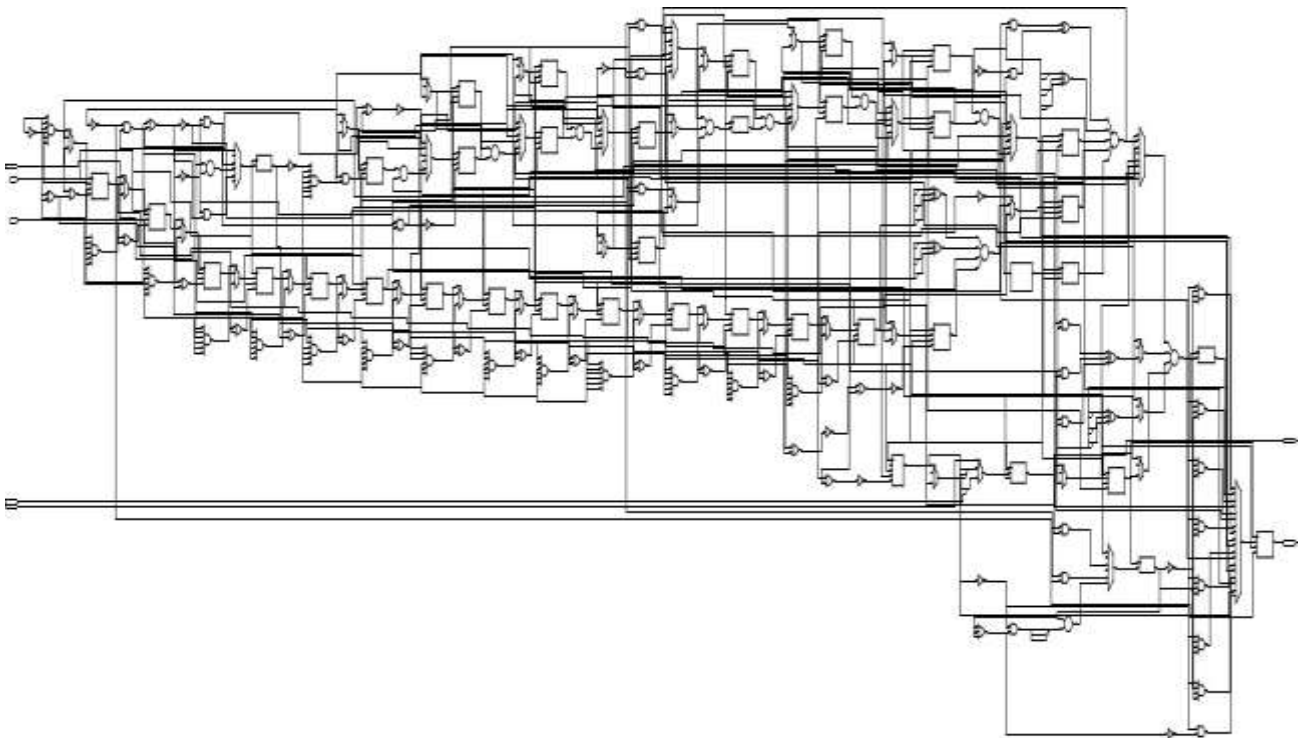


Fig.3 Schematic of SHA-256

5. CONCLUSION

The SHA-256 is chosen because of its complex security, high collision resistance, acceptable computation time along with its utilization in different applications. SHA-256 was used based on gated clock conversion, arithmetic resource sharing, and structural modelling of small building blocks. The proposed design along with the optimization techniques used to have resulted in a significant power and area reduction with a relatively large efficiency while maintaining a decent maximum frequency and throughput in comparison with other related work. The SHA-256 architecture processes a 512-bit block within 70 clock cycles. The message padding takes 3 cycle, hash computation takes 64 cycles, and outputting the final hash takes 1 cycle. The dynamic power consumption of the design is 14 mW. In this design the total area used in terms of slices is 505 (in comparison with 1373 slices) there is a 64% reduction. A throughput of 2659.4 Mbps is achieved which is 2.63 times larger than the previous design.

REFERENCES

1. M. Bahnasawi, A., K. Ibrahim, A. Mohamed, M. Khalifa, A. Moustafa, K. Abelmonim, Y. ismail, and H. Mostafa (2016), ASIC-Oriented Comparative Review of Hardware Security Algorithms for the Internet of Things Applications, IEEE International Conference on Microelectronics (ICM 2016), Cairo, Egypt, 285- 288.
2. N. Samir, A. S. Hussein, M. Khaled, A. N. ElZeiny, M. Osama, H. Yassin, A. Abdelbaky, O. Mahmoud, A. Shawky, and H. Mostafa (2019), ASIC and FPGA Comparative Study for IoT Lightweight Hardware Security Algorithms, Journal of Circuits, Systems, and Computers (JCSC), (Vol. 28), 1-13.
3. Q. Dang (2015), Changes in Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard, 69-73, <https://doi.org/10.1080/01611194.2012.687431>
4. M. Thakur (2018), Low Power Implementation of Secure Hashing Algorithm (SHA-2) using VHDL on FPGA of SHA-256, International Journal for Research in Applied Science and Engineering Technology, 2298-2303, (Vol. 6).
5. X. Guo, et al. (2010), On the Impact of Target Technology in SHA-3 Hardware Benchmark Rankings, IACR Cryptology ePrint Archive.
6. R. G. Dimond, et al (2006), Combining Instruction Coding and Scheduling to Optimize Energy in System-on-FPGA, 2006 14th Annual IEEE Symposium on Field Programmable Custom Computing Machines.
7. M. Thakur (2018), Low Power Implementation of Secure Hashing Algorithm (SHA-2) using VHDL on FPGA of SHA-256, International Journal for Research in Applied Science and Engineering Technology, 2298-2303, (Vol. 6).
8. C. Jeong and Y. Kim (2014), Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA, 2014 International SoC Design Conference (ISOCC).
9. B. S. Kaliski (1991), The MD4 message digest algorithm. In Advances in Cryptology—EUROCRYPT'90, Ivan Bjerre Damgård (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 492–492.
10. R. R. and S. Dusse. (1992), The MD5 Message-digest Algorithm. MIT Laboratory for Computer Science. [3] Xiaoyun Wang and Hongbo Yu. 2005. How to break MD5 and other hash functions. In Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 19–35.
11. X. Wang, Y. L. Yin, and H. Yu. (2005), Finding collisions in the full SHA-1. In Proceedings of the International Cryptology Conference (CRYPTO'05), Springer, 17–36, (Vol. 3621).
12. M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. (2017), The first collision for full SHA-1. In Proceedings of the International Cryptology Conference (CRYPTO'17). IACR, Santa-Barbara, United States, 570–596. DOI: http://dx.doi.org/10.1007/978-3-319-63688-7_19.
13. PUB FIPS. (2012) 180-2: Secure hash standard (SHS). US Department of Commerce, National Institute of Standards and Technology (NIST).
14. FIB PUB. (2008). 180-3, Secure hash standard (SHS). Federal Information Processing Standards Publication.
15. S. Chang, R. Perlner, W. E. Burr, M. S. Turan, J. M. Kelsey, S. Paul, and L. E. Bassham, (2012), Third-round report of the SHA-3 cryptographic hash algorithm competition. NIST Interag. Rep. 7896.
16. Q. Dang. (2013), Changes in federal information processing standard (FIPS) 180-4, secure hash standard. Cryptologia 37, 69–73.
17. M. J. Dworkin. (2015), SHA-3 Standard: Permutation-based Hash and Extendable-output Functions.
18. Technical Report. National Institute of Standards and Technology.
19. D. Thakur, (2018), Low Power Implementation of Secure Hashing Algorithm (SHA-2) using VHDL on FPGA of SHA-256.
20. S. B. Suhaili, J. Norhuzaimin, (2022), FPGA-based Implementation of SHA-256 with Improvement of Throughput using Unfolding Transformation.

21. Z. A. Al-Odat, M. Ali and A. Abbas and S. U. Khan, (2020), Secure Hash Algorithms and the Corresponding FPGA Optimization Techniques.
22. R. García, I Algreto-Badillo, M. Morales-Sandoval and C. Feregrino, (2013), A Compact FPGA-based processor for the Secure Hash Algorithm SHA-256.
23. S. Manickam, (2015), A study on performance study of enhanced SHA-256 algorithm.
24. S. Mishra, R. Jha, (2019), Low Power, and Simple Implementation of Secure Hashing Algorithm (SHA-2) Using VHDL implemented on FPGA of SHA-224/256 core.
25. H. Mestiri, F. Kahri, B. Bouallegue and M. Machhout, (2015), Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2.
26. V. Venkata, S. Karthik, and T. Venkata Sridhar (2013), Implementing sha-224/256 algorithm for secure commitment scheme applications using FPGA.
27. T. S Reddy, K.A. M Junaid, Y. Sukhi and Y. Jeyashree and P. Kavitha and V. Nath (2023), Analysis and design of wind energy conversion with storage system. e-Prime - Advances in Electrical Engineering, Electronics and Energy 100206(Vol. 17).
<https://doi.org/10.1016/j.prime.2023.100206>
28. D. Sharma, A. Rai, S. Debbarma, O. Prakash, M K Ojha and V. Nath (2023), Design and Optimization of 4-Bit Array Multiplier with Adiabatic Logic Using 65 nm CMOS Technologies, IETE Journal of Research, 1-14.
<https://doi.org/10.1080/03772063.2023.2204857>
29. J. Tirkey, S. Dwivedi, S. K. Surshetty, T. S. Reddy, M. Kumar, and V. Nath. (2023), An Ultra Low Power CMOS Sigma Delta ADC Modulator for System-On-Chip (SoC) Micro-Electromechanical Systems (MEMS) Sensors for Aerospace Applications. International Journal of Microsystems and Iot, 26–34 (Vol.1).
<https://doi.org/10.5281/zenodo.8186894>
30. D. Sharma, N. Shylashree, R. Prasad, and V. Nath. (2023), Analysis of Programmable Gain Instrumentation Amplifier. International Journal of Microsystems and Iot, 41–47(Vol. 1). <https://doi.org/10.5281/zenodo.8191366>
31. N. Anjum, V. K. Singh Yadav, and V. Nath. (2023). Design and Analysis of a Low Power Current Starved VCO for ISM band Application. International Journal of Microsystems and IoT, 82–98. (Vol. 1)
<https://doi.org/10.5281/zenodo.8288193>
32. K A Mohamed Junaid, Y Sukhi , N Anjum et.al., (2023). PV-based DC-DC buck-boost converter for LED driver. e-Prime - Advances in Electrical Engineering, Electronics

and Energy, 100271. (Vol. 5)
<https://doi.org/10.1016/j.prime.2023.100271>

AUTHORS



Hitesh M A received his B. Tech degree in electronics and communication engineering from B N M Institute of Technology Bengaluru, India in 2023. His areas of interest are VLSI Design, artificial intelligence, machine learning and human computer interaction.

Corresponding Author E-mail: hitesh.ajoy@gmail.com



Mahendra R received his B. Tech degree in electronics and communication engineering from B N M Institute of Technology Bengaluru, India in 2023. His areas of interest are VLSI Design, Data analytics, artificial intelligence, machine learning and robotics.

E-mail: rmahendra384@gmail.com



Sumanth H S received his B. Tech degree in electronics and communication engineering from B N M Institute of Technology Bengaluru, India in 2023. His areas of interest are VLSI Design, embedded system, machine learning and embedded system in IOT.

E-mail: sumanthhs15@gmail.com



Subodh Kumar Panda received his Bachelor of Engineering from Institution of Engineers (India) in 1996, Master of Technology from Visvesvaraya Technological University in 2000 and Doctor of Philosophy from Visvesvaraya Technological University in 2018. He is currently working as Professor in Department of Electronics and Communication Engineering, BNM Institute of Technology, Bengaluru. His main research work focuses on the development of sensors for green chilies. He is also interested in developing embedded solutions for portable applications and IoT based solutions. His areas of interest are VLSI and embedded systems.

E-mail: subodhpanda2013@gmail.com