Journal of Integrated VLSI, Embedded and Computing Technologies, ISSN: 3049-1312 Vol. 2, No. 2, 2025 (pp. 1-8)
RESEARCH ARTICLE ECEJOURNALS.IN

FPGA Hardware-Software Co-Design for Real-Time Embedded Systems

M.A. Erdoğan[1], F.N. Demir[2*]

[1.2]Department of Computer Engineering, Bogazici University, Istanbul, Turkey

Keywords:
FPGA Co-Design;
Real-Time Systems;
Embedded Systems;
Hardware-Software Integration;
Parallel Processing

Corresponding Author Email:
fatmadem.ir@boun.edu.tr

DOI: 10.31838/JIVCT/02.02.01

Received : 05.12.2024
Revised : 07.01.2025
Accepted : 03.03.2025

Abstract

Image processing, robotics, remote-sensing are many embedded systems which have critical real time performance requirement. Hence, such applications generally demand constraints on size, weight and power of a computing solution, for which traditional computing solutions fail to work. Field Programmable Gate Array (FPGAs) have emerged as a powerful platform to tackle these challenges through the hardware so software co design. Though this provides development people much more software flexibility, it allows one to do much higher performance work with custom hardware as well. In this paper we intend to venture the territory of HW/SW co plan based on FPGA for genuine time inserted frameworks and its favorable circumstances, strategy and genuine application. This will demonstrate that this approach drives a large improvement of embedded system performance while preserving the tradeoff between development time and system efficiency.

How to cite this article: Erdoğan MA, Demir FN (2025). FPGA Hardware-Software Co-Design for Real-Time Embedded Systems. Journal of Integrated VLSI, Embedded and Computing Technologies, Vol. 2, No. 2, 2025, 1-8

Introduction

Thus, Field Programmable Gate Array (FPGA) consists of a matrix composed of a configurable gate array matrix (CLB) interconnected through a programmable interconnect. FPGAs are programmable after manufacturing meaning logic circuit design can be achieved and the ability to almost infinite flexibility. Usually, the architecture of an FPGA includes: Elements that are configurable to perform the combinational and sequential logic functions. Routing resources such as interconnects that are programmable and connect logic blocks. Interfaces is a communication between external devices via I/O Blocks. Data storage and processing functions on chip RAM FPGAs are very good for prototyping and for small to medium production quantities because devices with this structure exist. The use of FPGAs in Embedded System has many benefits. Such embedded systems are good candidates for use in FPGAs as they are generally high performance, low power consumption, and capable of continual adaptation to changing requirements.[1-4]

Hardware Software Co Design Methodology

Hardware software co design implies that we must design the system in which the hardware and software are considered at the same time. The purpose of this methodology is to determine the amount of hardware and software implementation that provides the highest system performance.

Key principles for hardware-software co--design are: Using these principles, designers can successfully marry the attributes of the hardware side implementation to the software side implementation of embedded systems. The process of hardware-software co-design usually involves these following steps: The process enables designers to utilize benefits of both hardware and software implementations in the presence of the embedded system design problems. To support automatic synthesis of the hardware, hardware description languages (HDLs) must describe it with sufficient accuracy.[5-6]

Hardware Description Languages are such languages that are used with descriptions of digital

circuits. Among them, the two most popular HDLs for FIFA design are: VHSIC Hardware Description Language (VHDL): VHDL is an example of having strong typing and verbose syntax and is widely used in Europe and in the defense industry area. Verilog: As its C like syntax makes it a popular language among users, especially in the United States and Asia, semiconductor industry uses this language verilog. Now these two languages have the capability of describing hardware at various level of abstraction starting from low level of gate level description to the high level of behavioural description. This gap is overcome by high level so called synthesis tools; we want to have our developers describe hardware functionality on high level instead on low level bit by bit. Using Mentor Graphics Catapult HLSCo-Design for Real-Time Embedded Systems. Many embedded systems applications have real time performance as a critical requirement, image processing, robotics and remote sensing is just a few examples. This leads to requirements of these applications in terms of size, weight and power consumption that are beyond the scope of traditional computing solutions. The challenges mentioned above have been addressed by Field-Programmable Gate Arrays (FPGAs) as a powerful platform for hardware-software co-design. By using this approach, developers create embedded solutions that use the speed of custom hardware and the flexibility of software.

This article presents an outlook on FPGA based hardware-software co-design for real time embedded systems with its benefits, associated methodologies and some real world applications. We will take a closer look into the details of such an approach, and how it can improve embedded system performance by a large factor while also balancing development time with efficiency of the system (Figure 1).[7-9]

Understanding FPGA Technology

Semiconductor device Field Programmable Gate Arrays (FPGAs) are matrix of configurable logic blocks (CLB) connected by programmable interconnects. In contrast to Application Specific Integrated Circuits (ASICs), FPGAs can be reprogrammed post manufacturing giving design flexibility that is unmatched with digital circuit design. Configurable elements that can implement various combinational and sequential logic functions are logic blocks. This makes FPGAs ideally suitable for prototyping and small to medium scale production runs as owing to unique structure developers can implement custom hardware designs to meet application criteria.[10]

Co-simulation tools allow the designer to validate the functionality of the hardware software system that has been designed from abstract view of actual hardware. Some popular co-simulation environments include: In addition to other things, these tools give a more thorough test of the co designed system through waveform viewing, assertion based verification, and coverage analysis. Some problems with real time image processing on embedded systems are:[11]

1. Therefore, invariably, image processing algorithms involve a lot of computational powers on large datasets.
2. Data Preprocessing for High Resolution Image: the data is required to have large memory bandwidth as well as resource efficient data management.

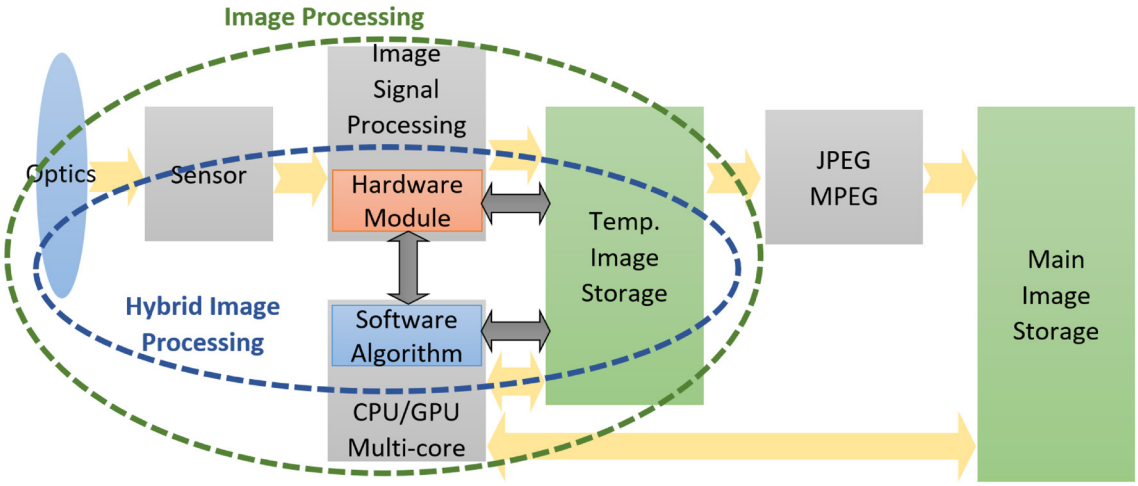


Fig. 1: Understanding FPGA Technology

3. Energy efficient implementations are important because nowadays embedded systems have limited power budget.
4. High Frame Rate: Processing at high frame rate is required for many applications and extremely low (or zero) latency.
5. Usually, Algorithms are updateable or different processing modes are supported.

FPGA's parallel processing capabilities, the customizability of their memory hierarchies and the reconfigurability make FPGAs well suited to solve these problems. Acceleration of Image Processing Algorithms on an FPGA. FPGA acceleration can provide several image processing algorithms (Figure 2):

Convolution based filters like Gaussian blur or edge detection can be written as filtering task, so engineers can be sure that the implementation will be efficient because convolutions are highly parallelizable on FPGAs. Efficient Implementation of transformations from RGB to YUV space in Hardware. The technology accelerates Image Feature Algorithms such as SIFT or SURF, therefore Custom Hardware can greatly speedup feature extraction. JPEG or H.264 codecs can be used to support real time encoding and decoding using FPGA parallelism via compression/decompression. Specialized hardware structures can be used to perform erode, dilate and other morphological operations. Hence, we demonstrate that FPGAs can be pipelined to induce large speedups over software only solutions and will provide real time performance in embedded systems.[12-14]

Image Processing Memory Management

Consequently, high performance image processing on FPGAs demands efficient memory These methods make it possible to balance trade offs between how much on chip memory is used, how much bandwidth is required to main memory, and how fast the processing needs to be. I then provide a case study in FPGA hardware software co design for a robot localization system which works in real time. The system processes the images from a camera mounted on ceiling of the arena about the positions and motions of multiple robots in the arena. We can achieve up to 10 robots by simultaneous detection and tracking and the position accuracy is sub centimeter; we are able to run at 25 frames per second on 1600x1200 pixel images. Furthermore, low power consumption makes it possible to verify the integrated hardware-software system in an embedded form factor before implementation on the actual hardware. Some popular co-simulation environments include (Figure 3):[15]

The features of these tools often include waveform viewing, assertion based verification and coverage analysis to verify that the co-designed system has been built properly and meets all requirements. These challenges are well suited to be addressed by using FPGAs, which provide parallel processing

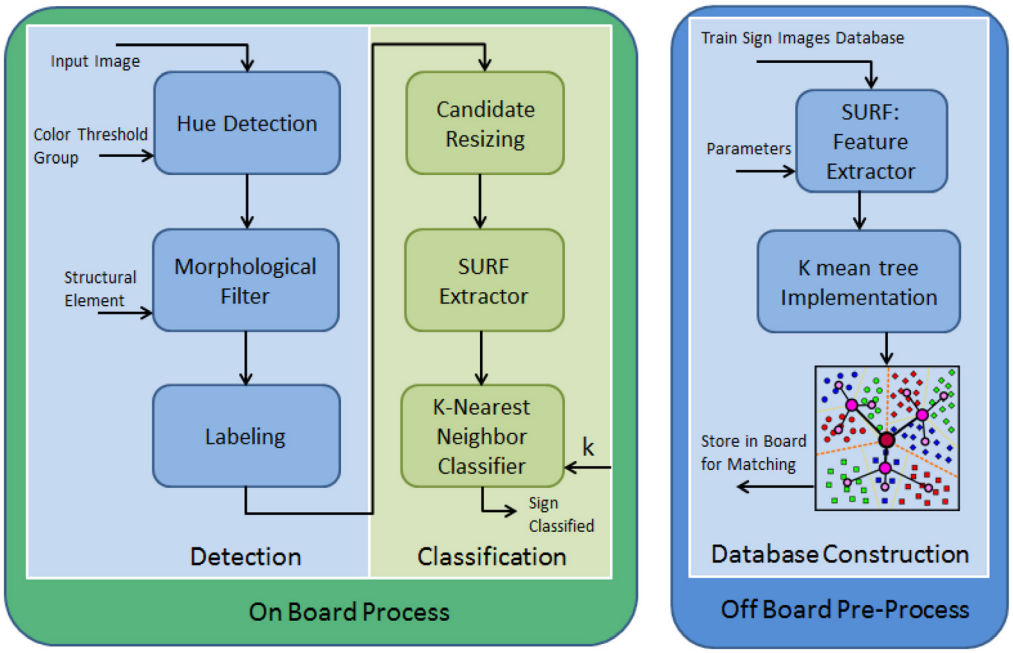


Fig. 2: FPGA's parallel processing

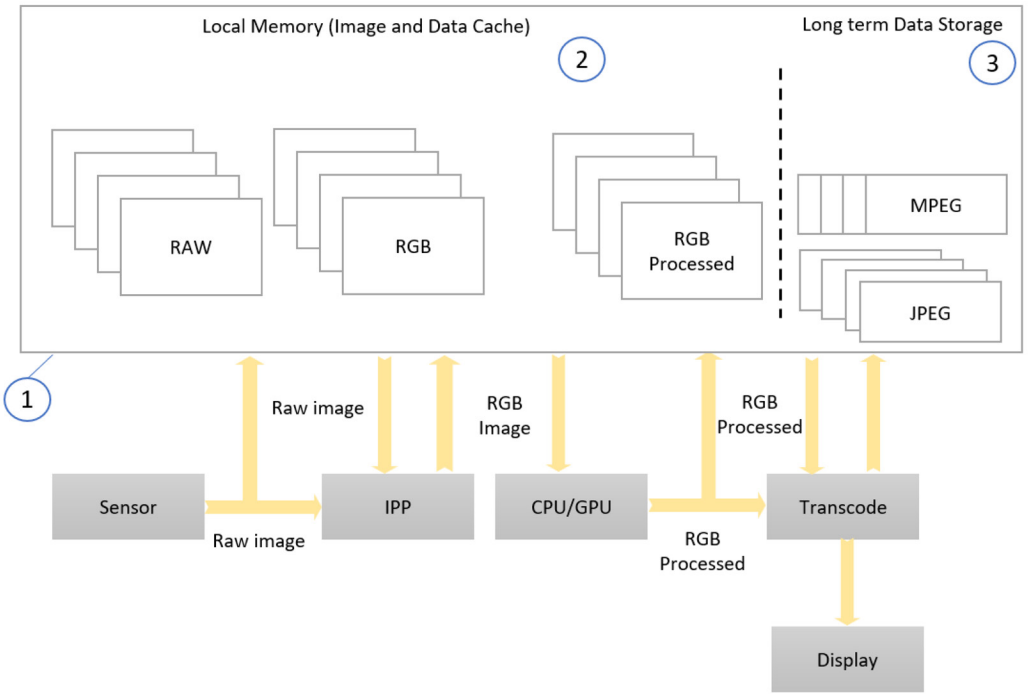


Fig. 3: Image Processing Memory Management

capabilities, user configurable memory hierarchies and/or reconfigurability.. FPGA-Accelerated Image Processing Algorithms. FPGA acceleration can be applied to solve several image processing algorithms. Filtering: Convolution based filters such as Gaussian blur or edge detection can be parallelized very well on FPGAs. Efficient Implementation of Color Space Conversion: Transforming from the color space RGB to YUV is fairly simple in hardware. Accelerating algorithms like SIFT or SURF for Feature Extraction since it is based on image features detection. Real time encoding and decoding with FGPAs can be taken advantage of using JPEG or H.264 Cocecs. Morphological operations (Erosion, dilation and so on) can be carried using specialized hardware structures. With the implementation of these algorithms in the hardware, FPGAs achieve significant speedups with respect to the realizations of these algorithms in the software thus allowing real time performance in embedded systems.[16-18]

Memory Management for Image Processing

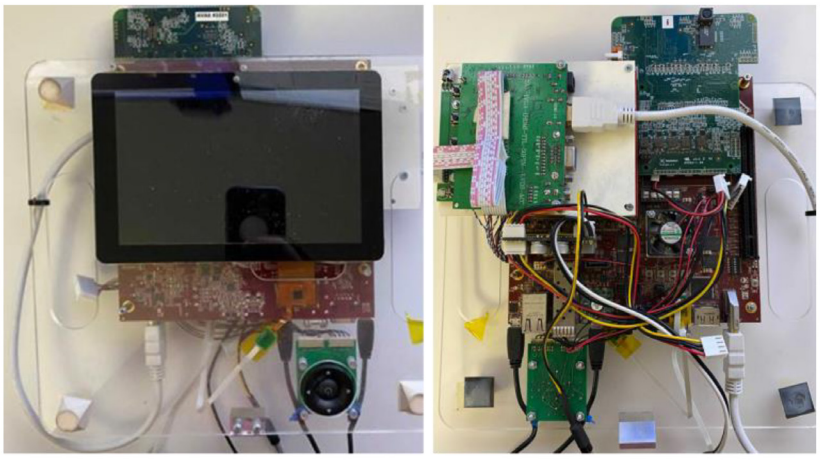
High performance FPGA image processing requires efficient memory management. Strategies include: Storing a few lines of the image in on chip memory so as to reduce the number of external memory access. Dividing the image into smaller tiles that can be independently and in parallel processed. Dual buffers are utilized to overlap data transfer and processing operations (Ping-Pong Buffers). Caching: Optimization of both algorithm and basic data structures. Burst Transfers: Making maximum use of available bandwidth of external memory interfaces through burst modes. The use of these techniques allows designers to best balance the trade offs of bandwidth of external memory, on chip memory usage, and processing efficiency.

In the following, we discuss a case study on an FPGA based hardware software co design for a real time robot localization system. Images of robots in an arena can be obtained from a ceiling mounted camera and passed for tracking the position of multiple robots to the system. Hardware-Software Partitioning. 3. Image acquisition and pre processing. Binary image of greyscale image generation by using color thresholding 4.The connected component labeling 5. Detected blobs centroid calculation 6. Robot identification and tracking 7. Mapping and coordinate transformation 8.Robot and external systems communication 9. System control, user interface. Designers can use co simulation tools to verify the functionality of the integrated hardware software system before fabricated into the actual hardware. Some popular co-simulation environments include:[19]

These challenges can be addressed by using FPGAs owing to their parallel processing capabilities, customizable memory hierarchies and their reconfigurability. FPGA-Accelerated Image Processing Algorithms. Some of these algorithms may be accelerated by using FPGA. Filtering: Convolution based filtering such as Gaussian blur or edge detection can be made highly parallel on CPU and FPGAs. Color Space Conversion: We can perform fast hardware computations in transforming color spaces (e.g., RGB to YUV). Feature Extraction: It is possible to accelerate algorithms such as SIFT or SURF to find image features by using custom hardware. JPEG or H.264 codecs: There is real time encoding and decoding capacity for compression/decompression using FPGAs in parallel. Morphological Operations: Specified hardware structures can be used to implement Erosion, dilation and other morphological operations. These algorithms are implemented in hardware and by doing so they can provide large speedups over software only solutions which would allow for true real-time performance in embedded systems. Memory management for high-performance image processing on FPGAs is crucial. Strategies include. Storing a couple of lines of the image in on chip memory to reduce external memory access (Line Buffers). Divide the image into small tiles that can be processed independently and in parallel. Dual Buffers: Overlapped data transfer and procesing operations using dual buffers. Caching: writing custom cache structures to achieve the access patterns of a specific algorithm. Burst Transfers Policy: Making use of burst modes of external memory interfaces to ensure maximum bandwidth usage. These techniques support the designer in the tradeoffs between on chip memory usage, external memory bandwidth, and processing efficiency (Fogure 4).

Thus, we will consider FPGA based hardware-software co-design case study of a real time robot localization system. The system we describe tracks the positions of several robots in an arena, where images come from a ceiling mounted camera. The FPGA parallel processing capability is exploited for low level computationally intensive, but forced image processing tasks, while the flexibility of software is used for more higher level choices and system management. Direct pixel streaming from an interface to custom image sensor. Parallel processing pipelines for color thresholding, and pipeline for generating a binary image. Sometimes it is necessary to apply optimized novel connected component labeling algorithm for FPGA implementation. Ease of development and easy connectivity with embedded Linux operating system and a pipelined fixed point arithmetic along with a centroid calculation (Table 1).[20-21]

Embedded to lessen the development and connectivity is a simple embedded Linux operating system. It is a multi threaded application for running multiple tasks concurrently. An interfacing communication protocol based on TCP/IP between robots and external system An implementation of the hardware-software system functionality before actual hardware implementation. Some popular co-simulation environments include: Such a package as HDL simulation tool, ModelSim, works with both VHDL and Verilog. Questa is an advanced verification tool that can be used for mixed language simulation and



PoC front side PoC back side

Fig. 4: Experimental set-up

Table 1: FPGA Metrics

Metric	Hardware (FPGA)	Software (Processor-based)
Execution Time (Latency)	Lower latency, faster execution	Higher latency, slower execution
Power Consumption	Low power usage due to parallelism	Higher power consumption, CPU-bound
Throughput	High throughput	Lower throughput
Development Time	Longer due to hardware design	Shorter with high-level software
Flexibility/Adaptability	Limited, fixed design	Very flexible, adaptable code

Table 2: FPGA Contribution

Evaluation Metric	FPGA Contribution	Software Contribution
Deadline miss rate	FPGA provides deterministic performance	Software may introduce variability
System scalability, load balancing	FPGA enables scalability by offloading parallel tasks	Software adapts to different hardware set-ups
Response time	FPGA handles critical tasks quickly	Software may add overhead in handling I/O
Resource usage percentage (CPU, memory, FPGA LUTs)	FPGA reduces CPU resource load by performing parallel tasks	Software requires CPU resources but can be optimized

formal verification. Xilinx Vivado Simulator: Integrated simulator in the Xilinx Vivado Design Suite. Such tools often also offer features such as waveform viewing, assertion based verification and coverage analysis so as to guarantee testing of the entire system that is co designed ((Table 2).[22-25]

Challenges in Embedded Image Processing

There are a few challenges imposed by real time image processing on the embedded systems which are:

1. Image processing algorithms typically consist of complex mathematical operations on large dataset, thus image processing is computationally expensive.
2. **Data Bandwidth:** This deals with the fact that you have to deal with high resolution images which means you are going to need a significant amount of memory bandwidth as well as be able to manage your data.
3. Given that Embedded systems usually have strict power budgets, implementations are required to be power efficient.
4. Many applications involve performing image processing real time and at high frame rate with low latency.
5. They also have to be adaptable in order to update algorithms or be able to support multiple processing modes.

These challenges are well suited to be addressed by FPGAs, which are able to perform parallel processing, customize memory hierarchies and have the ability to reconfigure.

Conclusion

Based on hardware software co design process using FPGA based hardware, it is a good method to develop hardware software which is targeted for real time high performance embedded system software. In conjuction with the flexibility of software and the acceleration properties of custom hardware, solution are created that can be efficient to meet the stringent application requirements in image processing, robotics and the like. As a case study of a robot localization system, we show that this approach can be applied practically, it significantly outperforms software only solutions in all aspects except power consumption, and the resulting form factor remains compact and power efficient. Embedding based on FPGA technology along with new trends towards a heterogeneous computing, and the emergence of the new technologies 3D FPGA and adaptive computing, are increasingly embedding more future proof embedded systems. However, in order to achieve full potential from FPGA based co design, one must cope with design complexity, energy efficiency and productivity. Using the top end tools, optimizing designs from low to high levels, and timely alert to the trends of the surface, developers are enabled to create breakthrough embedded systems for real

time processing and control. Embedded systems in FPGA are poised for playing key roles to accelerate technological advancements in crucial areas like edge computing and 5G communications, autonomous systems, and space applications. Engineers with mastery of art and science of hardware software co design have tremendous capability to unlock new possibilities and drive innovation in rapidly expanding world of embedded systems.

References:

1. Bhattacharyya, S. S., Eker, J., Janneck, J. W., Lucarz, C., Mattavelli, M., & Raulet, M. (2009). Overview of the MPEG reconfigurable video coding framework. Journal of Signal Processing Systems, (July). https://doi.org/10.1007/s11265-009-0399-3
2. Raulet, M., Urban, F., Nezan, J. F., Moy, C., Deforges, O., & Sorel, Y. (2006). Rapid prototyping for heterogeneous multicomponent systems: An MPEG-4 stream over a UMTS communication link. EURASIP Journal on Applied Signal Processing, Article ID 64369.
3. Gund, A., et al. (2008). Design evaluation of a home-based telecare system for chronic heart failure patients. IEEE Engineering in Medicine and Biology Society, 5851–5854.
4. Villalba, et al. (2009). Wearable and mobile system to manage remotely heart failure. IEEE Transactions on Information Technology in Biomedicine, 13(6), 990–996.
5. Yao, J., et al. (2005). A wearable point-of-care system for home use that incorporates plug-and-play and wireless standards. IEEE Transactions on Information Technology in Biomedicine, 9(3), 363–371.
6. Benini, L., Bruni, D., Drago, N., Fummi, F., & Poncino, M. (2002). Virtual in-circuit emulation for timing accurate system prototyping. ASIC/SOC Conference, 2002. 15th Annual IEEE International, 49–53.
7. Pittala, C. S., et al. (2021). Novel methodology to validate DUTs using single access structure. 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, September 24–25, 1–5.
8. Koehler, C., Mayer, A., & Herkersdorf, A. (2008). Chip hardware-in-the-loop simulation (CHILS) - Embedding microcontroller hardware in simulation. Proceedings of the 19th IASTED International Conference on Modelling and Simulation.
9. Nageldinger, U., Pyttel, A., & Kleve, H. (2004). System simulation speedup combining SystemC models and reconfigurable hardware. Retrieved from http://speac.fzi.de/WORKSHOP2/SpeacParis2004_01.pdf
10. Kuskin, J., et al. (1994). The Stanford FLASH multiprocessor. 21st Annual International Symposium on Computer Architecture.
11. Mok, A. K. (1984). The design of real-time programming systems based on process models. Real-Time Systems Symposium.
12. Park, C. Y. (1992). Predicting deterministic execution times of real-time programs (PhD thesis). University of Washington, Technical Report 92-08-02, Department of Computer Science & Engineering.
13. Babu, P. A., Nagaraju, V. S., & Vallabhuni, R. R. (2022). 8-bit carry look ahead adder using MGDI technique. In IoT and Analytics for Sensor Networks (pp. 243–253). Springer, Singapore.
14. Oniga, S., Tisan, A., Mic, D., Buchman, A., & Vida-Ratiu, A. (2008). Optimizing FPGA implementation of feed-forward neural networks. Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM 2008), Brasov, Romania, May 22–23, 31–36.
15. Tisan, A., Oniga, S., Buchman, A., & Gavrincea, C. (2007). Architecture and algorithms for synthesizable neural networks with on-chip learning. International Symposium on Signals, Circuits and Systems (ISSCS 2007), Iasi, Romania, July 12–13, 1, 265–268.
16. Riccobene, E., & Scandurra, P. (2009). Weaving executability into UML class models at PIM level. First European Workshop on Behaviour Modelling in Model Driven Architecture (BM-MDA), Enschede, The Netherlands, 10–28.
17. Mallet, F., Andre, C., & DeAntoni, J. (2009). Executing AADL models with UML/MARTE. International Conference on Engineering of Complex Computer Systems, Germany, 371–376.
18. Silva-Filho, A. G., et al. (2011). An ESL approach for energy consumption analysis of cache memories in SoC platforms. International Journal of Reconfigurable Computing, 2011, 1–12.
19. Bhargava, G. U., Midasala, V., & Vallabhuni, R. R. (2022). FPGA implementation of hybrid recursive reversible box filter-based fast adaptive bilateral filter for image denoising. Microprocessors and Microsystems, 90, 104520.
20. Parno, B., & Perrig, A. (2005). Challenges in securing vehicular networks. Proceedings of Workshop on Hot Topics in Networks (HotNets-IV), November.
21. Golle, P., Greene, D., & Staddon, J. (2004). Detecting and correcting malicious data in VANETs. VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, 29–37.
22. Jiang, K., Eles, P., & Peng, Z. (2011). Optimization of message encryption for distributed embedded systems with real-time constraints. 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS '11), April, 243–248.
23. Andrews, D., Niehaus, D., & Ashenden, P. (2004). Programming models for hybrid FPGA/CPU computational components. IEEE Computer, January.
24. Bazargan, K., Kastner, R., Ogrenci, S., & Sarrafzadeh, M. (2000). A C to hardware-software compiler.

Retrieved from http://www.ece.ucsb.edu/~kastner/papers/C_to_hardware_software_FCCM00

25. Mejail, M., Nestares, B. K., & Gravano, L. (2024). The evolution of telecommunications: Analog to digital. Progress in Electronics and Communication Engineering, 2(1), 16–26. https://doi.org/10.31838/PECE/02.01.02
26. Muralidharan, J. (2024). Optimization techniques for energy-efficient RF power amplifiers in wireless communication systems. SCCTS Journal of Embedded Systems Design and Applications, 1(1), 1-6. https://doi.org/10.31838/ESA/01.01.01
27. Prasath, C. A. (2024). Energy-efficient routing protocols for IoT-enabled wireless sensor networks. Journal of Wireless Sensor Networks and IoT, 1(1), 1-7. https://doi.org/10.31838/WSNIOT/01.01.01
28. Muralidharan, J. (2024). Innovative materials for sustainable construction: A review of current research. Innovative Reviews in Engineering and Science, 1(1), 16-20. https://doi.org/10.31838/INES/01.01.04
29. Rahim, R. (2024). Optimizing reconfigurable architectures for enhanced performance in computing. SCCTS Transactions on Reconfigurable Computing, 1(1), 11-15. https://doi.org/10.31838/RCC/01.01.03
30. Prasath, C. A. (2024). Optimization of FPGA architectures for real-time signal processing in medical devices. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 11-15. https://doi.org/10.31838/JIVCT/01.01.03
31. Rahim, R. (2023). Effective 60 GHz signal propagation in complex indoor settings. National Journal of RF Engineering and Wireless Communication, 1(1), 23-29. https://doi.org/10.31838/RFMW/01.01.03
32. Alnumay, W. S. (2024). Use of machine learning for the detection, identification, and mitigation of cyber-attacks. International Journal of Communication and Computer Technologies, 12(1), 38-44. https://doi.org/10.31838/IJCCTS/12.01.05
33. Yeonjin, K., Hee-Seob, K., Hyunjae, L., & Sungho, J. (2023). Venting the potential of wirelessly reconfigurable antennas: Innovations and future directions. National Journal of Antennas and Propagation, 5(2), 1–6.