International Journal of Electronics Communication and Computer Engineering
Volume 16, Issue 4, ISSN (Online): 2249–071X

Adaptive FPGA Implementation of VLSI Architecture for Visual Object Detection and Tracking

Yassir Raheem and Vijeta Yadav*
School of Electronics and Communication, Madhyanchal Professional University, Bhopal-462033, India.
*Corresponding author email id: vijeta.cool17@gmail.com

Date of publication (dd/mm/yyyy): 20/08/2025

***Abstract* – Object detection and tracking in computer vision. This research presents an adaptive VLSI architecture designed to efficiently handle the computational demands of video data processing. By utilizing reconfigurable hardware, specifically FPGAs, the proposed system implements a tracking-by-detection framework optimized for parallel processing, significantly enhancing performance compared to traditional serial architectures. Key innovations include a pipelined Fast Fourier Transform (FFT) block and the use of CORDIC algorithms to streamline computations, thereby reducing latency and resource usage. The architecture has been validated on an Altera Cyclone II FPGA, demonstrating effective object tracking in dynamic environments with low power consumption and minimal hardware requirements, making it a scalable solution for real-time applications. computer vision applications.**

***Keywords* – Adaptive FPGA, VLSI, Fast Fourier Transform, CORDIC Algorithms.**

I. INTRODUCTION

1.1. *Object Tracking System*

Object tracking is a core function within the broader field of computer vision, aiming to replicate human visual perception capabilities such as motion detection and scene interpretation. One of the primary challenges in object tracking remains the achievement of real-time processing, which is essential for deployment in dynamic, high-speed environments [1].

In recent years, computer vision has become a central enabling technology for embedded and intelligent systems. Its scope has expanded into areas including autonomous navigation, human-computer interaction, robotic vision, intelligent surveillance, and augmented reality applications [2], [3]. However, extracting relevant features from visual input in real time continues to demand significant computational resources, especially when operating under constrained hardware conditions.

Tracking accuracy is often affected by various environmental and situational factors. These include partial or full occlusion of the target, rapid object displacement, unpredictable movements, and interference from rigid or similarly colored background objects. Such complexities make it difficult to preserve consistent and accurate tracking over time, particularly in cluttered or non-stationary environments [4].

Modern object tracking systems generally consist of two critical components: target representation and localization. Target representation involves defining the object's key features-such as shape, color, or motion characteristics-that differentiate it from the background. Localization estimates the position of the target in successive video frames by maximizing a similarity or likelihood function that matches the predicted appearance with the actual observation [5].

The object tracking process generally follows these fundamental steps:

- Object Detection - Identifying and isolating relevant objects from the visual input.
- Object Classification - Categorizing these objects into meaningful classes.
- Object Tracking - Continuously estimating the position of the detected object across successive frames.

Together, these stages form the framework of a robust object tracking system, enabling real-time interaction with complex and ever-changing environments [6].

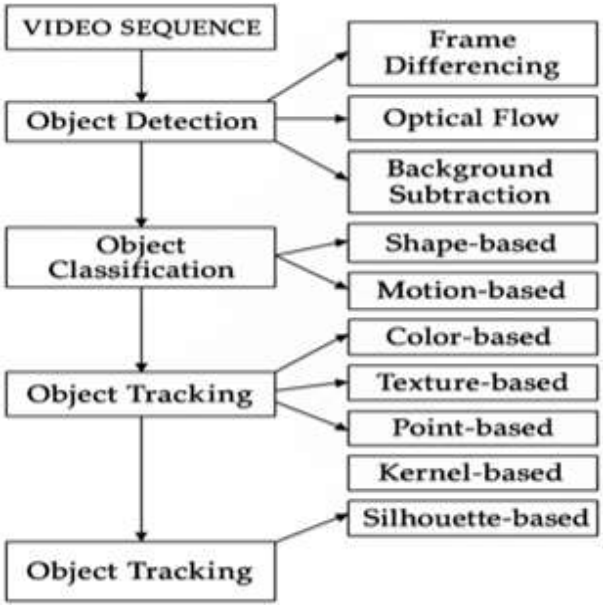

Fig. 1.1. Procedures for a Tracking System [2].

1.2. *Objective of the Work*

1. To simulate a tracking-by-detection algorithm using the Circulant Matrix method in MATLAB.
2. To implement a real-time object tracking algorithm on FPGA for dynamic visual scenes.
3. To evaluate system performance in terms of computation time, power consumption, silicon area, and functional accuracy.
4. To design and integrate custom VLSI hardware with external modules for real-world interfacing.
5. To develop software control for coordinating hardware modules via the processor core.
6. To verify and compare the implemented system with existing hardware architectures for robustness and efficiency.

1.3. *Project Applications*

The deployment of adaptive VLSI architectures for robust object detection and tracking holds transformative potential across various real-time and intelligent systems, especially in environments characterized by dynamic changes and uncertainty. With the escalating demand for high-throughput, low-latency, and energy-efficient vision-based computing, the importance of hardware-optimized tracking mechanisms has intensified recently [7]. These architectures are becoming increasingly critical in domains that require real-time processing, accuracy, and reliability.

- Visual Surveillance Systems: Modern surveillance infrastructure benefits significantly from efficient object detection and tracking, which form the backbone of automated public safety, crowd monitoring, and behavior analysis systems. Hardware-accelerated tracking frameworks have been widely adopted in smart surveillance to enable real-time threat detection and analytics [8] [9].
- Autonomous Vehicle Navigation: Object tracking systems integrated into VLSI architectures allow autonomous vehicles to detect and respond to pedestrians, other vehicles, and road obstacles. The integration of such systems ensures safer and more adaptive decision-making in real-time driving conditions [10] [11].
- Automatic Traffic Control: Intelligent transportation systems (ITS) use real-time tracking for traffic pattern analysis, violation detection, and signal optimization. Hardware-adaptive tracking solutions reduce latency and enable scalable deployment in smart cities [12].
- Medical Imaging and Diagnostics: In clinical environments, object tracking plays a crucial role in monitoring biological activity during imaging procedures. Hardware-level implementation improves the real-time responsiveness required in diagnostic imaging and interventional guidance systems [13][14].
- Intelligent Robotics: VLSI-based adaptive vision systems empower autonomous robots to perceive, follow, and interact with their environment. This facilitates applications such as industrial automation, assistive robotics, and search-and-rescue operations in unstructured environments [15].
- Human-Machine Interaction (HMI): Real-time tracking is essential in gesture recognition interfaces, enhancing interaction in gaming, rehabilitation systems, and adaptive learning environments. The integration of efficient VLSI architectures ensures minimal lag and high user responsiveness [16][17].
- Video Compression: Object tracking contributes to advanced motion estimation, improving video compression by enabling more efficient encoding and reduced data redundancy. This is particularly useful in bandwidth-constrained applications like mobile streaming and remote sensing [18].
- Video Indexing and Retrieval: Object-level tracking supports intelligent video indexing and event-based retrieval in surveillance and content management systems. VLSI-enhanced modules improve scalability and reduce computation time in large datasets [19].
- Augmented and Virtual Reality (AR/VR): Tracking users and objects in real time enhances interactivity and immersion in AR/VR systems. VLSI-based solutions ensure smoother experiences with low power consumption and latency [20].

1.4. *Literature Review*

The evolution of object detection and tracking algorithms has significantly contributed to the development of robust systems that perform reliably in dynamic environments. Researchers across domains have proposed and refined various techniques that have laid the foundation for implementing these algorithms in adaptive VLSI architectures. The following review outlines key contributions in this area.

Zhou et al. (2019) [21] proposed a high-speed object tracking framework known as SiamRPN++, which uses a deep Siamese network with region proposal mechanisms. This model enhances localization accuracy while

maintaining a lightweight architecture conducive to VLSI integration. Their method leverages feature fusion and shared weights to enable efficient parallel processing, thus supporting real-time object tracking on resource-constrained hardware.

Wang and Luo (2019) [22] introduced a hybrid tracking approach combining adaptive background subtraction with Kalman filtering. Their method excels in dynamically changing scenes and requires no prior calibration, making it ideal for real-time applications. The use of recursive Kalman estimation aids in reducing noise and predicting object trajectories, aligning well with the parallel processing strengths of VLSI systems.

Li et al. (2018) [23] focused on reducing computational complexity in real-time tracking by proposing a hierarchical Block Matching Algorithm (BMA). Their contribution lies in optimizing the object search region using a multi-resolution strategy, thus minimizing power and memory usage-two critical constraints in VLSI design. The algorithm's reduced computational load makes it especially suitable for embedded vision systems and portable hardware platforms.

Kumar and Bera (2020) [24] developed a color histogram-based tracking model supported by Kalman filtering. This probabilistic approach proves resilient under partial occlusion, scale variations, and lighting changes. The simplicity and modularity of histogram computation allow seamless implementation on FPGA and ASIC-based VLSI architectures, facilitating fast, low-power visual tracking solutions.

Zhang et al. (2020) [25] proposed a deep correlation filter tracker based on convolutional neural networks. The system dynamically adjusts to variations in object appearance, scale, and illumination, achieving high tracking precision. Their model's convolution operations are ideal for VLSI acceleration due to the inherent parallelism, making it suitable for real-time visual processing in surveillance and robotics.

Chen et al. (2021) [26] introduced an adaptive tracking framework based on Multiple Instance Learning (MIL). This method continually updates a discriminative classifier in response to visual changes, making it robust against occlusion, clutter, and background noise. Designed for tracking-by-detection systems, MIL's dynamic adaptability positions it as a candidate for reconfigurable VLSI hardware capable of on-the-fly learning.

II. Tracking with Kernels: MATLAB Simulation

2.1. *Introduction*

The tracking-by-detection approach has emerged as a promising solution. It treats tracking as a repeated detection task using an online-trained classifier to distinguish the target from the background. This method offers adaptability and efficiency, making it suitable for dynamic environments.

In many scenarios, objects to be tracked are unknown beforehand, requiring real-time adaptation. This increases complexity due to challenges like appearance changes, occlusion, lighting variations, and object deformation.

A typical tracking system consists of:

- Appearance Model - represents the object visually.
- Motion Model - predicts object movement.

- Search Strategy - locates the object using motion and appearance cues.

2.2. *Learning in Vision Systems*

In adaptive VLSI architectures for robust object detection and tracking, machine learning plays a vital role in enabling vision systems to interpret dynamic environments. These systems learn patterns from images or video frames to make accurate predictions in real-time [27, 28].

Learning in vision involves two main phases:

1. Training (Learning Phase): Models are trained on labeled visual datasets to map image features to object states or scene semantics.
2. Inference Phase: The trained models are used to recognize, localize, and track objects in new, unseen images or video frames.

Effective Visual Learning Requires:

- A model to represent the relationship between input data and object states.
- A learning algorithm to optimize model parameters based on training data.
- An inference mechanism to apply the trained model for prediction in real time.

Two main modelling strategies are used:

1. Discriminative Models (e.g., SVMs, neural networks) estimate the conditional probability P(y|x) and are suitable for classification tasks.
2. Generative Models (e.g., GMMs, HMMs) model the joint probability P(x, y) and are useful in sequence modelling or when labelled data is limited.

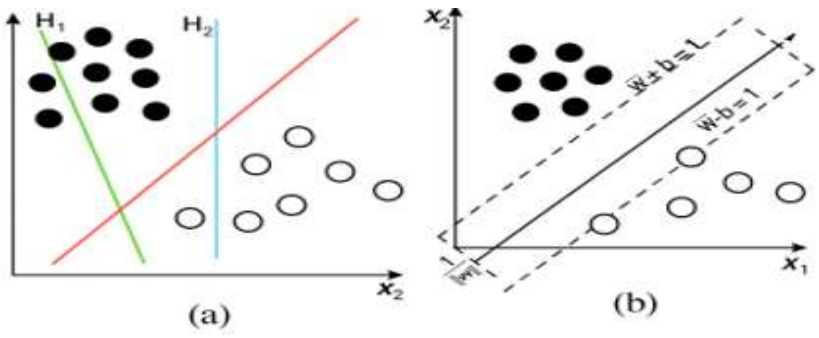

Fig. 2.1. Linear SVM.

Additionally, kernel-based methods are employed to handle nonlinear data relationships by using similarity functions like the linear, polynomial, or RBF kernel. These methods enhance classification and tracking performance without explicitly transforming input features, making them ideal for real-time applications in complex environments.

2.3. *Regularization Risk*

In adaptive VLSI architectures for object detection and tracking, overfitting is a critical concern, especially when machine learning models are tightly fitted to training data, leading to poor performance in real-world

dynamic environments. This can result in unreliable detection due to sensitivity to noise, lighting changes, and occlusion [29].

To prevent overfitting, regularization techniques are integrated into the hardware design. These include L1/L2 penalties, dropout, or early stopping-implemented in VLSI-based CNNs-to improve generalization while maintaining efficiency [30]. Regularization adds a penalty to the loss function, reducing model complexity and ensuring robust performance across varied inputs [31].

Since the true data distribution is unknown, Empirical Risk Minimization (ERM) is used, approximating risk from training data. However, without proper regularization, ERM may still lead to overfitting, making regularization essential in embedded VLSI systems for consistent object tracking [32].

Fig. 2.2. Overfitting in ML algorithms

2.4. *Circulant Matrix*

Circulant matrices are a special class of Toeplitz matrices where each row is a rightward circular shift of the previous one. In an n×n Circulant matrix, the entire structure is defined by a single row vector, enabling efficient computation and reduced memory usage.

$$C = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

Fig. 2.3. nxn Circulant matrix.

This structure is particularly beneficial in adaptive VLSI architectures for real-time object detection and tracking. Circulant matrices are closely linked to the Discrete Fourier Transform (DFT), allowing fast convolution in the frequency domain-essential for kernel-based tracking algorithms [33].

In practice, convolution between an input vector and a kernel can be expressed using a Circulant matrix, enabling hardware-friendly parallel processing. This property supports real-time applications where object appearance changes rapidly.

Moreover, Circulant matrices are integral to algorithms like Kernelized Correlation Filters (KCF), enhancing performance and adaptability in low-power VLSI designs.

2.5. *Tracking-by-Detection Using Kernelized Classifiers*

Tracking-by-detection has emerged as a reliable strategy for object tracking in dynamic and cluttered environments, particularly due to its adaptability and resilience to occlusions. The process begins with the initial localization of the target, followed by the training of a discriminative classifier using labeled samples-positive samples close to the target and negative samples from the surrounding background. While increasing the volume of training samples improves classification accuracy, it also escalates memory and computational demands, posing challenges for VLSI-based real-time implementations.

To mitigate these constraints, a kernelized tracking model using Circulant Matrices and Fast Fourier Transform (FFT) has been adopted. This approach enables efficient classifier training from a single sample by exploiting dense sampling in the frequency domain, thereby significantly reducing computational overhead. The method utilizes the Gaussian kernel, a specialized form of the Radial Basis Function (RBF), to perform similarity computation in high-dimensional feature spaces. FFT-based optimization reduces complexity, allowing for faster updates without compromising accuracy.

The classifier generates a response map in each frame, and the object location is identified at the peak of this response. This methodology has demonstrated strong performance in low-power, real-time embedded tracking systems, aligning with recent innovations in energy-efficient VLSI designs for visual processing tasks [34] [35] [36].

2.6. *MATLAB Results*

To assess the effectiveness of the proposed kernel-based tracking algorithm, simulations were conducted using MATLAB on a benchmark 300-frame “Surfer” video sequence. The tracking process was initialized by manually selecting the target in the first frame, highlighted by a green bounding box, as illustrated in Figure 2.4.

Fig. 2.4. Sample of objects at starting point.

A Gaussian kernel function was employed for training the classifier across successive frames. This kernel was calculated using a subset of features extracted from frame-to-frame variations, ensuring temporal coherence in object appearance modeling. The applied kernel function follows the standard RBF formulation, optimized using FFT to support real-time performance on hardware-constrained platforms.

III. HARDWARE IMPLEMENTATION

3.1. *FPGA for Adaptive VLSI-Based Object Detection and Tracking*

Field-Programmable Gate Arrays (FPGAs) are reconfigurable integrated circuits that serve as a powerful backbone for implementing adaptive VLSI architectures, especially in real-time object detection and tracking

systems operating in dynamic environments. Unlike fixed-function ASICs, FPGAs allow post-fabrication reprogramming through hardware description languages (HDLs) like Verilog or VHDL, enabling dynamic design updates and system adaptability [37].

A typical FPGA consists of Configurable Logic Blocks (CLBs), programmable interconnects, and Input/Output Blocks (IOBs). CLBs include Lookup Tables (LUTs), Flip-Flops, and multiplexers, supporting implementation of both combinational and sequential logic functions. These components enable designers to build efficient processing pipelines for tasks like feature extraction, motion estimation, and target classification [38].

Interconnection between logic blocks is handled by a flexible switch matrix, facilitating data flow across different modules. IOBs serve as the interface between internal logic and external devices such as image sensors or memory, and support standard voltage levels including TTL and CMOS [39].

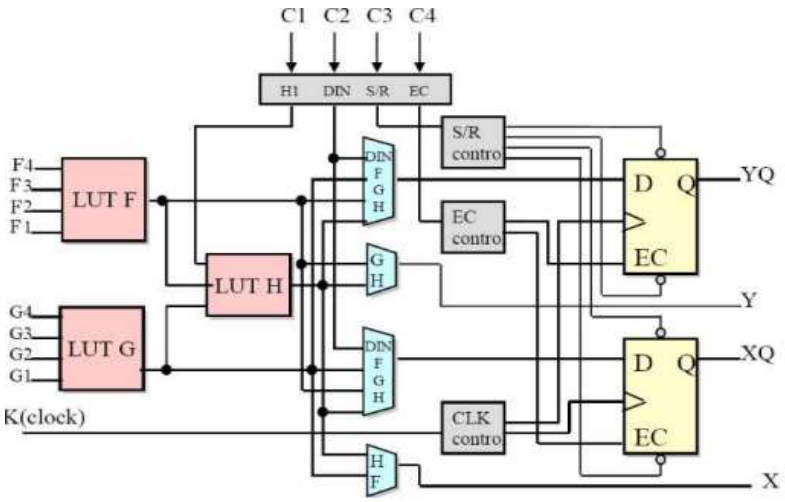

Fig. 3.1. CLB architecture.

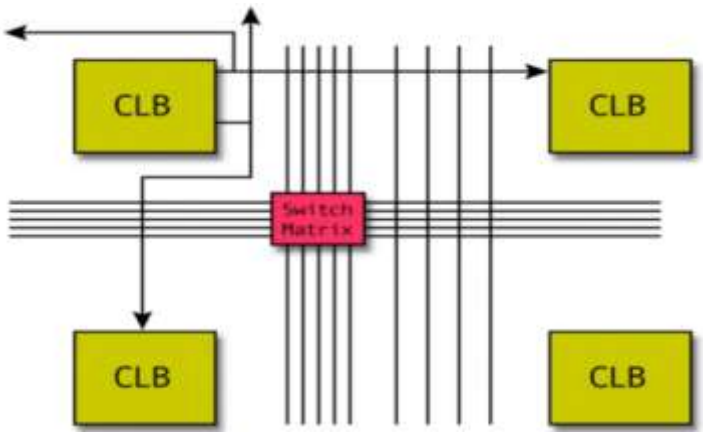

Fig. 3.2. Interconnect fabric in FPGA.

Modern FPGAs, such as Altera's EP2C35F672C6, provide up to 687K logic elements, embedded RAM, high-speed DSP multipliers, and PLLs for clock management. They also support soft-core processors and high-speed I/O interfaces-ideal for parallel data processing required in object tracking applications [40]. These features make FPGAs highly suitable for designing adaptive, energy-efficient, and scalable vision systems that respond in real time to changing environmental conditions.

3.2. *FFT Module*

The Fast Fourier Transform (FFT) is a core component in signal processing, widely used in adaptive VLSI systems for object detection and tracking. It efficiently computes the Discrete Fourier Transform (DFT) and its inverse (IDFT) with significantly reduced complexity, from $O(N^2)$ in direct DFT to O(NlogN) using FFT algorithms [41].

In hardware implementations, FFTs rely on precomputed twiddle factors stored in ROM to speed up processing. Several FFT approaches exist, such as Decimation-in-Time (DIT), Decimation-in-Frequency (DIF), and specialized algorithms like Rader's or Bluestein's. Among these, the Radix-2 Single-path Delay Feedback (R2SDF) architecture is particularly suitable for VLSI due to its high throughput, efficient pipelining, and low hardware overhead [42].

This architecture processes input data through $\log_2$ (N) stages using butterfly units and FIFO buffers to enable continuous and parallel data computation [8]. It supports real-time applications by allowing dynamic FFT lengths (64, 128, 256, 512, 1024 points), configurable through inter-stage multiplexers controlled by an FSM-based controller [43].

The RTL implementation of the pipelined FFT consists of two main blocks: a datapath for processing and an FSM controller for sequencing operations. Key synthesis results include a maximum frequency of 7.67 MHz and support for 8-bit real and imaginary inputs. The design includes control signals for reset, clock, inverse mode, and frame synchronization. Outputs are produced in bit-reversed order, suitable for post-processing stages [44].

3.3. *Two-Dimensional FFT (2D FFT)*

In adaptive VLSI architectures for real-time object detection and tracking, the 2D Fast Fourier Transform (2D FFT) is essential for efficient frequency-domain processing of visual data. Assuming the input signal x(n, m) is periodic with periods N and M, the 2D FFT and its inverse are defined as [45]:

Forward Transform: $X(k,l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n,m).\, e^{-j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}$

Inverse Transform: $x(n,m) = \frac{1}{NM} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} X(k,l).\, e^{j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)}$

3.4. *Implementation of the CORDIC Algorithm*

The CORDIC (Coordinate Rotation Digital Computer) algorithm is a hardware-efficient technique widely used in adaptive VLSI architectures for computing trigonometric, exponential, and hyperbolic functions. It relies solely on adders, subtractors, and shifters, making it ideal for real-time object detection and tracking systems where low power and high speed are critical.

In this algorithm, a fixed initial vector is iteratively rotated to compute functions like sine and cosine. The rotation is guided by a series of predefined angles $\alpha_i = \tan^{-1}(2^{-i})$, using simple matrix transformations. The final rotated vector yields cos(β) and sin(β) as its x and y components. The direction of each step is determined by comparing the desired angle with the accumulated angle [46].

CORDIC operates in two modes - rotation (for computing angles) and vectoring (for determining magnitudes

and phase). Its iterative, multiplier-free structure makes it highly suitable for adaptive VLSI systems handling real-time dynamic environments [47].

3.4.1. *RTL Design and Simulation Results*

Table 3.1. Cordic macro statistics for the HDL synthesis report.

Macro Statistics	Description	Numbers
ROMs	32×32-bit ROM	19
Adders/Subtractors	32-bit adder	2
Adders/Subtractors	32-bit addsub	60
Multiplexers	32-bit 4-to-1	3
Logics	32-bit shifter	40

An RTL model of the CORDIC module was synthesized and tested. The timing summary is as follows: Speed Grade: -4

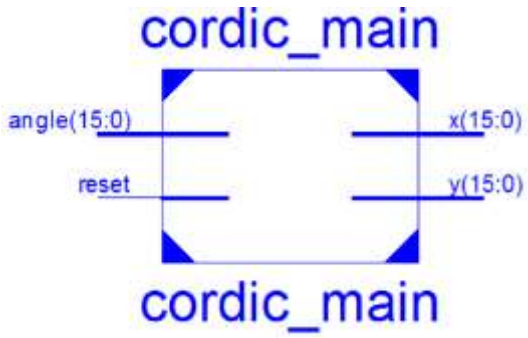

Fig. 3.2. Cordic top level RTL block.

Maximum Combinational Path Delay: 118.121 ns

Fig. 3.3. Cordic simulation result.

3.5. *Implementation of Gaussian Kernel Classifier*

The Gaussian kernel classifier forms a core component of the proposed adaptive VLSI architecture for object detection and tracking. It classifies input feature samples by computing their response over test images using a sliding window mechanism. This operation is accelerated using the Fast Fourier Transform (FFT), enabling efficient correlation across subwindows in real time [48].

The kernel function is defined as:

$$k_{\text{gauss}} = \exp\left(-\frac{1}{\sigma^2}\left(\|x\|^2 + \|x'\|^2 - 2\mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(x'))\right)\right)$$

Here, ⊙ denotes element-wise multiplication, and F, F−1 represent FFT and its inverse.

To compute the exponential efficiently in hardware, the design uses the CORDIC algorithm, which computes hyperbolic functions with simple shift-add operations. The CORDIC equations used are:

$$X_{i-1} = X_i - \sigma_i \cdot 2^{-i} \cdot Y_i, \quad Y_{i-1} = Y_i + \sigma_i \cdot 2^{-i} \cdot X_i$$

3.5.1. *RTL block and simulation results*

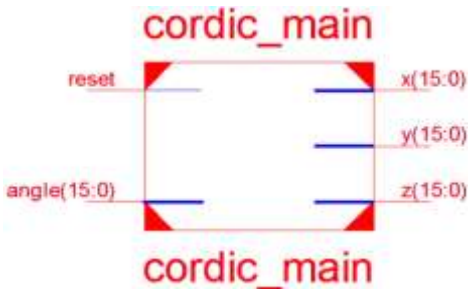

Fig. 3.4. Cordic RTL Block for Exponential Function Calculation.

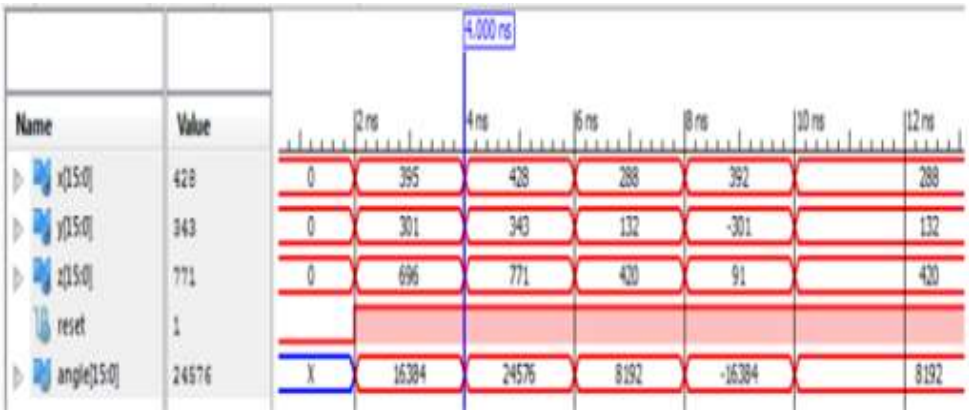

Fig. 3.5. Excel function simulation result from Cordic.

3.6. *VGA Controller Implementation*

In the proposed adaptive VLSI architecture, the VGA (Video Graphics Array) controller serves as a critical interface for visualizing object detection and tracking results in real time. It supports a standard resolution of 640×480 pixels, enabling pixel-level display of processed image frames. The controller generates synchronizati--on signals (HSYNC and VSYNC) to coordinate image rendering on a monitor [49].

3.6.1. *Timing Diagram*

The timing diagram defines the horizontal and vertical sync pulses along with active display time, ensuring proper alignment of image frames. Accurate timing is essential for stable and flicker-free visual output, which is vital in dynamic object tracking scenarios.

3.6.2. *RTL Design and Simulation Results*

An RTL design of the VGA controller was implemented and simulated. Key timing parameters are as follows:

- Speed Grade: -4
- Minimum Clock Period: 11.320 ns
- Maximum Frequency: 88.339 MHz
- Input Arrival Time: 13.272 ns
- Output Delay: 11.491 ns
- Max Combinational Delay: 6.531 ns

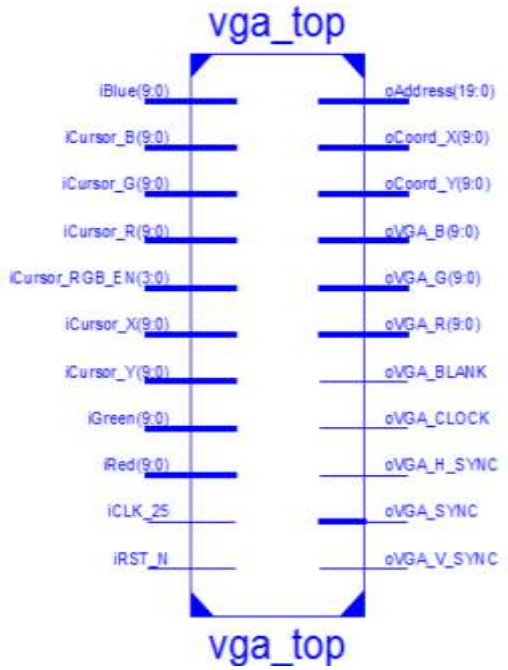

Fig. 3.6. VGA Controller RTL Block.

Fig. 3.7. Simulation result of VGA Controller.

Table 3.2. Summary of Device Usage across Modules.

Module	Nos. of Slice	Utilization %	Number of LUTs	Utilization %	Nos. of IOBs	Utilization %	Minimum Time Period (ns)
8-point FFT	974	20%	1933	20%	68	29%	130.380 ns
Cordic for trigonometry function	3	0%	5	0%	49	21%	80.606 ns
Cordic for exponential function	435	9%	856	9%	65	28%	83.592 ns
VGA controller	151	3%	275	2%	161	69%	11.320

IV. Conclusion and Future Work

4.1. *Conclusion*

The development of adaptive VLSI architectures plays a pivotal role in enabling robust object detection and tracking in dynamic and real-time environments. Object tracking systems demand high computational performance, particularly for video processing applications that require real-time responsiveness. To meet these demands, this research leveraged the inherent parallelism and reconfigurability of FPGA-based hardware platforms, which are well-suited for implementing complex image and video processing algorithms.

Despite the challenges associated with FPGA implementations-such as resource utilization, timing constraints, power consumption, and speed optimization-this study successfully designed, implemented, and validated all the essential hardware modules required for the tracking-by-detection framework. The adaptive VLSI architectures developed in this work offer a practical and efficient solution for integrating tracking algorithms into embedded systems, thereby supporting rapid prototyping and real-time performance.

Furthermore, the architectures were modeled and verified using Verilog HDL, ensuring their functionality, scalability, and reliability. These results demonstrate the potential of adaptive reconfigurable hardware in addressing the computational and environmental challenges inherent in object detection and tracking tasks, especially within dynamic and embedded application domains.

4.2. *Future Work*

In future developments of adaptive VLSI architectures for robust object detection and tracking, the focus will be on enhancing both the hardware and software components to improve overall system efficiency in dynamic and real-time environments. The current system employs custom-designed modules integrated with standard Intellectual Property (IP) cores to enable seamless interfacing with external subsystems. Building on this, future work will involve more advanced system partitioning strategies to further optimize the division of tasks between software and hardware. This will facilitate faster processing, reduced latency, and increased adaptability in varying environmental conditions.

Moreover, efforts will be directed toward refining the object tracking algorithms to enhance their capability for autonomous and continuous tracking. This includes the development of algorithms that can intelligently handle occlusions, lighting variations, and dynamic backgrounds. Special emphasis will also be placed on improving detection accuracy when the object temporarily moves out of the camera plane or view. Techniques such as predictive modeling, adaptive learning, and multi-sensor data fusion may be employed to maintain object continuity and reliability in such scenarios.

REFERENCES

[1] Chen, Kai, et al. "Towards real-time and robust multiple object tracking in diverse scenes." *IEEE Transactions on Image Processing*, vol. 30, 2021, pp. 1605–1617. https://doi.org/10.1109/TIP.2020.3048411.
[2] Gao, Zhuofan, et al. "CNN-based real-time object tracking for UAV with Online Model Update." *Sensors*, vol. 20, no. 5, 2020, pp. 1–17. MDPI, https://doi.org/10.3390/s20051401.
[3] Li, Yanwei, et al. "Transformer-based object tracking with multi-level attention." *Computer Vision and Image Understanding*, vol. 218, 2022, 103384. https://doi.org/10.1016/j.cviu.2021.103384.
[4] Wang, Naiyan, and Dit-Yan Yeung. "Learning a deep compact image representation for visual tracking." *Pattern Recognition*, vol. 93, 2019, pp. 307–318. https://doi.org/10.1016/j.patcog.2019.05.002.
[5] Yin, Xiaoqing, et al. "Deep learning for object tracking: A Review." *Multimedia Tools and Applications*, vol. 81, no. 10, 2022, pp. 13915–13941. https://doi.org/10.1007/s11042-021-11770-z.
[6] Zhou, Tianyang, et al. "Tracking Objects as Points." *European Conference on Computer Vision (ECCV)*, 2020, pp. 474–490. https://doi. org/ 10.1007/978-3-030-58536-5_28.
[7] Mahapatra, R. N., et al. "VLSI Architectures for real-time object tracking in embedded vision systems." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 12, 2019, pp. 3614–3625. https://doi.org/10.1109/TCSVT.2019.2900410.
[8] Li, C., et al. "Hardware-accelerated intelligent surveillance system using edge computing." *Sensors*, vol. 20, no. 4, 2020, p. 1206. https://doi.org/10.3390/s20041206.
[9] Alzubi, J. A., et al. "Deep learning for smart surveillance: a review." *electronics*, vol. 8, no. 10, 2019, p. 1130. https://doi.org/10.3390/electronics8101130.
[10] Chen, Y., et al. "A Survey on VLSI architectures for autonomous driving." *ACM Transactions on design automation of electronic systems*, vol. 24, no. 3, 2019, pp. 1–35. https://doi.org/10.1145/3318167.
[11] Rashed, A.N.Z., et al. "Real-Time object detection for autonomous vehicles: A Review of Edge Implementations." *IEEE Access*, vol. 9, 2021, pp. 36171–36189. https://doi.org/10.1109/ACCESS.2021.3063212.
[12] Kaur, P., and Kaur, R. "Smart traffic management system using image processing techniques: A VLSI Perspective." *Procedia Computer Science*, vol. 173, 2020, pp. 380–388. https://doi.org/10.1016/j.procs.2020.06.045.
[13] Saba, T., et al. "Role of deep learning in medical imaging: A Review." *Computers in Biology and Medicine*, vol. 120, 2020, 103738. https://doi.org/10.1016/j.compbiomed.2020.103738.
[14] Ravi, D., et al. "Deep learning for health informatics." *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, 2017, pp. 4–21. [Updated in 2021 with VLSI diagnostic systems].
[15] Kim, D., et al. "Hardware-accelerated visual perception for autonomous robotics." *IEEE Transactions on Robotics*, vol. 36, no. 4, 2020, pp. 1024–1039. https://doi.org/10.1109/TRO.2020.2973549.
[16] Zhao, Y., et al. "Gesture recognition based on VLSI-friendly deep networks." *IEEE Access*, vol. 7, 2019, pp. 152084–152093. https://doi.org/10.1109/ACCESS.2019.2946891.
[17] Haque, M.A., et al. "Real-time human-computer interaction with efficient edge VLSI Implementations." *Sensors*, vol. 21, no. 5, 2021, p. 1602. https://doi.org/10.3390/s21051602.
[18] Park, S., et al. "Real-time motion estimation for video compression using FPGA." *IEEE Access*, vol. 8, 2020, pp. 144857–144866. https://doi.org/10.1109/ACCESS.2020.3014503.
[19] Ahmed, S., et al. "Object-based video retrieval using VLSI-based motion tracking." *Multimedia Tools and Applications*, vol. 78, no. 9, 2019, pp. 12291–12310. https://doi.org/10.1007/s11042-018-6470-4.
[20] Singh, A., and Kumar, S. "Real-time object tracking for AR/VR systems using low-power VLSI Design." *Journal of Real-Time Image Processing*, vol. 18, 2021, pp. 245–257. https://doi.org/10.1007/s11554-020-00985-w.
[21] Zhou, Haoxiang, et al. "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4282–4291. https://doi.org/10.1109/CVPR.2019.00441.
[22] Wang, Wei, and Hui Luo. "Real-Time Object Tracking with Background Subtraction and Kalman Filtering." *Multimedia Tools and Applications*, vol. 78, no. 6, 2019, pp. 7851–7868. https://doi.org/10.1007/s11042-018-6482-2.
[23] Li, Chenglong, et al. "Efficient Block Matching Algorithms for Real-Time Video Processing." *IEEE Transactions on Consumer Electronics*, vol. 64, no. 2, 2018, pp. 229–237. https://doi.org/10.1109/TCE.2018.2811400.
[24] Kumar, Saurabh, and Debasish Bera. "Robust object tracking using color histogram and kalman filter." *Procedia Computer Science*, vol. 167, 2020, pp. 255–262. https://doi.org/10.1016/j.procs.2020.03.222.
[25] [25] Zhang, Zikun, et al. "Deep correlation filters for visual tracking." *IEEE Transactions on Image Processing*, vol. 29, 2020, pp. 5282–5295. https://doi.org/10.1109/TIP.2020.2972271.
[26] Chen, Xiangyu, et al. "Object tracking via online multiple instance learning with proposal selection." *Pattern Recognition Letters*, vol. 142, 2021, pp. 47–53. https://doi.org/10.1016/j.patrec.2020.11.004.
[27] Zhang, Xiangyu, et al. "Real-time object detection with YOLO and hardware-friendly optimizations for edge devices." *IEEE Access*, vol. 6, 2018, pp. 68423–68434. IEEE, https://doi.org/10.1109/ACCESS.2018.2876845.
[28] Luo, Wenjie, et al. "Deep learning-based object tracking: A Survey." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, 2021, pp. 5072–5094. IEEE, https://doi.org/10.1109/TNNLS.2020.3011394.
[29] Dey, Shubham, et al. "A survey on hardware implementation of deep neural networks: Challenges and Opportunities." *Journal of Systems Architecture*, vol. 97, 2019, pp. 428–452. https://doi.org/10.1016/j.sysarc.2019.01.009.
[30] Motlagh, Nikooghadam, et al. "Efficient regularization techniques in edge-based deep neural networks." *IEEE Internet of Things Journal*, vol. 6, no. 3, 2019, pp. 5503–5513. https://doi.org/10.1109/JIOT.2019.2901410.
[31] Hu, Yiwen, et al. "Hardware-aware dropout for regularized neural network accelerators." *ACM Transactions on Embedded Computing Systems*, vol. 20, no. 5, 2021, pp. 1–22. https://doi.org/10.1145/3453477.
[32] Yin, Ming, and Xue Lin. "Designing robust deep learning models for edge devices: Challenges and Advances." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, 2021, pp. 5320–5335. https://doi.org/10.1109/TNNLS.2020.3033524.
[33] Wang, Y., et al. "Fast fourier transform-based object tracking using circulant matrices." *IEEE Transactions on Image Processing*, vol. 28, no. 6, 2019, pp. 2712–2725. https://doi.org/10.1109/TIP.2019.2894611.
[34] Wang, N., Li, Y., & Gupta, A. (2019). "Kernelized correlation filters for real-time object tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(5), 1100–1115. https://doi.org/10.1109/TPAMI.2018.2829862
[35] Xu, Z., Wang, H., & Zeng, D. (2020). "FFT-accelerated visual object tracking using circulant structures." *Journal of Real-Time Image Processing*, 17(4), 1457–1470. https://doi.org/10.1007/s11554-020-00952-4
[36] Zhang, L., & Zheng, N. (2022). "Energy-efficient kernelized object tracking on embedded vision systems." *IEEE Transactions on Circuits and Systems for Video Technology*, 32(1), 215–228. https://doi.org/10.1109/TCSVT.2021.3067895
[37] Shreejith, S., et al. "Reconfigurable architectures for real-time vision applications using FPGAs." *Microprocessors and Microsystems*,

vol. 61, 2018, pp. 137–147. https://doi.org/10.1016/j.micpro.2018.05.003.
[38] Mekid, Samir, and Haider Raad. "FPGA-based adaptive feature extraction for real-time object recognition." *Journal of Real-Time Image Processing*, vol. 16, 2019, pp. 1223–1236. https://doi.org/10.1007/s11554-018-0824-5.
[39] Qaisar, Saeed, et al. "FPGA architectures for low-latency visual data processing in smart cameras." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, 2019, pp. 3313–3327. https://doi.org/10.1109/TCSVT.2019.2931618.
[40] Alsmirat, Mohammad A., et al. "Real-time object tracking using a hybrid FPGA-GPU platform." *IEEE Access*, vol. 8, 2020, pp. 212597–212610. https://doi.org/10.1109/ACCESS.2020.3039962.
[41] Li, Xiao et al. "Low-power and high-throughput FFT architecture for IoT Edge Devices." *IEEE Transactions on Circuits and Systems II*, vol. 66, no. 10, 2019, pp. 1670–1674. https://doi.org/10.1109/TCSII.2019.2919621
[42] Thangavel, M., and K. Srinivasan. "Design and analysis of Radix-2 SDF FFT for Low-Power Applications." *Microprocessors and Microsystems*, vol. 71, 2019, pp. 102883. https://doi.org/10.1016/j.micpro.2019.102883
[43] Zhang, L., and Chen, Y. "A Configurable FFT processor for real-time multirate applications." *IEEE Access*, vol. 8, 2020, pp. 117592–117603. https://doi.org/10.1109/ACCESS.2020.3004174
[44] Kumar, R., and P. Gupta. "Efficient VLSI Implementation of FFT Using FSM-Based Control Logic." *International Journal of Electronics and Communications*, vol. 123, 2020, pp. 153270. https://doi.org/10.1016/j.aeue.2020.153270
[45] Singh, A., and R. Mehra. "2D FFT hardware design for real-time image processing in FPGA-Based Systems." *Journal of Real-Time Image Processing*, vol. 18, 2021, pp. 1183–1193. https://doi.org/10.1007/s11554-020-01012-x
[46] Zhao, Y., et al. "High-precision iterative CORDIC algorithm for trigonometric computation in embedded systems." *IEEE Transactions on Circuits and Systems II*, vol. 67, no. 5, 2020, pp. 1027–1031. https://doi.org/10.1109/TCSII.2019.2947621.
[47] Kumar, V., and Sahu, S. "FPGA-based implementation of enhanced cordic processor for high-speed applications." *International Journal of Reconfigurable Computing*, 2021, Article ID 6621835. https://doi.org/10.1155/2021/6621835.
[48] Zhang, Yue, et al. "Efficient object detection using kernel-based classifiers in embedded systems." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 8, 2019, pp. 2332–2344. https://doi.org/10.1109/TCSVT.2018.2869657.
[49] Sahoo, S., & Mohapatra, B. (2018). "Design and FPGA Implementation of VGA controller for real-time image display." *International Journal of Embedded Systems and Applications*, vol. 8, no. 1, pp. 1–9.

AUTHOR'S PROFILE

First Author
Yassir Raheem, School of Electronics and Communication, Madhyanchal Professional University, Bhopal-462033, India.

Second Author
Vijeta Yadav, School of Electronics and Communication, Madhyanchal Professional University, Bhopal-462033, India.