

Concept For Using Fpga-Based Computer Vision In Smart-Home IoT Paradigm

Leonid Poskotin

School of Electronic Engineering, Tianjin University of Technology and Education, Tianjin 300222, China

Email: svp_vpl@yahoo.com

How to cite this paper: Poskotin, L. (2026). Concept For Using Fpga-Based Computer Vision In Smart-Home IoT Paradigm. Academic Journal of Emerging Technologies, 2(3), 63-70. ISSN Print: 3104-4417; ISSN Online: 3104-4425.

<https://doi.org/10.63313/AJET.9047>

Published: 2026-04-30

Copyright © 2026 by author(s) and Erytis Publishing Limited.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Abstract

The smart home IoT sector continues to grow every year, with an ever-increasing number of new solutions emerging that incorporate computer vision technologies using CNN-based machine learning approaches. For the most part, CNN computations are performed either on a cloud server—which can increase computation time to 100 milliseconds and leave the data vulnerable to uncontrolled cyberattacks—or on edge devices powered by expensive GPUs or low-performance CPUs. This article describes a concept for an FPGA-based device built on the ZYNQ platform, which, in theory, could resolve these issues. The proposed concept is capable of capturing video feed using an OV5640 camera, processing it, and performing object detection using a CNN model. The system can manage and control most types of IoT devices using the Home Assistant software, installed as a Docker container on a custom-built Linux distribution. As part of this paper, an experiment was conducted to compare the inference times of an FPGA-based and a CPU-based CNN accelerator on YOLOv3 model. According to the results of the experiment, the FPGA-based solution performed best. This result could serve as a foundation for future work on integrating FPGAs into the IoT paradigm.

Keywords

Internet of Things; Field-programmable Gate Array; Computer Vision

1. Introduction

In today reality the Internet of Things (IoT) paradigm has become an integral part of daily life for residents of major cities, particularly in the field of home automation. By IoT influence many ordinary objects become “smart” devices with capabilities of involving in local and worldwide nets. In retail stores, “smart” LED bulbs—which can be controlled via mobile devices—sit on the shelves alongside regular light bulbs, and “smart” robot vacuum cleaners clean people’s homes on a scheduled basis. Various sensors, including cameras and edge devices, can capture data for activating various types of actuators using different scripts.

AI and related technologies are increasingly being integrated into all areas of human sphere and activity. From 2016 to 2023, the annual number of AI-related publications doubled (from 100,000 to 244,000 per year), Global private investment in this sector increased from approximately \$10 billion to \$37 billion between 2023 and 2024.

In this regard user activates and activate the corresponding scripts, where EL-HARP video recognition function is based on gradient-boosted decision tree ML method.

Many solutions that combine IoT and ML are based on the “cloud computing” paradigm, which means that the primary ML processing of data collected on-site via sensors takes place on a centralized server located remotely from the site, which have big disadvantages. One of main from them is a high latency time from sending input data and receiving processed output data from cloud server, that sometimes can range from 10 milliseconds to 100 milliseconds or more. Another is a privacy and security issues, because many cloud servers are deployed in public big data centers across the world, which can become targets for massive cybersecure attacks, from which malicious third-party can obtain access to sensitive data from IoT sensors that stored on cloud server storage, that responsibility for securing the system falls primarily on the data center owner, which largely prevents the owner of the IoT system from ensuring adequate security through their own efforts. That issues can be solved by implementing paradigm of edge computing, that insist on placing ML calculations on local resource-constrained devices. Mostly local resource-constrained devices presented by slow subscalar computation power Central Processing Unit (CPU) based solutions or fast but expensive Graphics Processing Unit (GPU) based solutions. These devices can run the ML model’s computation on the device to save energy by keeping data without sending them constantly to the Cloud , which reduces the risk of sensitive data to be stolen and ensures data privacy. However, deploying traditional ML models on such devices is challenging due to limited computational resources, memory, hardware specifications, and power consumption constraints. The presented concept design is this paper is strive to solve these problems by presenting Field-programmable gate array (FPGA) as a core device for making IoT smart home system with computer vision capabilities.

2. Hardware design

2.1. Core device structure

The proposed system consists from two main components: core device and Smart Home IoT device network. The recommended hardware configurations represent the minimum system requirements for performing the specified functions; they may be replaced with equivalent models from other manufacturers or with higher-performance versions.

Core device is a host device that on charge of establishing and maintaining IoT

network and its functionality. Its role is to collecting data using different sensors from both itself and the IoT network, processing of this input data using CNN Deep learning methods with using output data result as a trigger condition for controlling of IoT network via scripts and scenarios. Core device is represented by Advanced Micro Devices (AMD) Xilinx Zynq UltraScale MPSoCs EG series FPGA model.

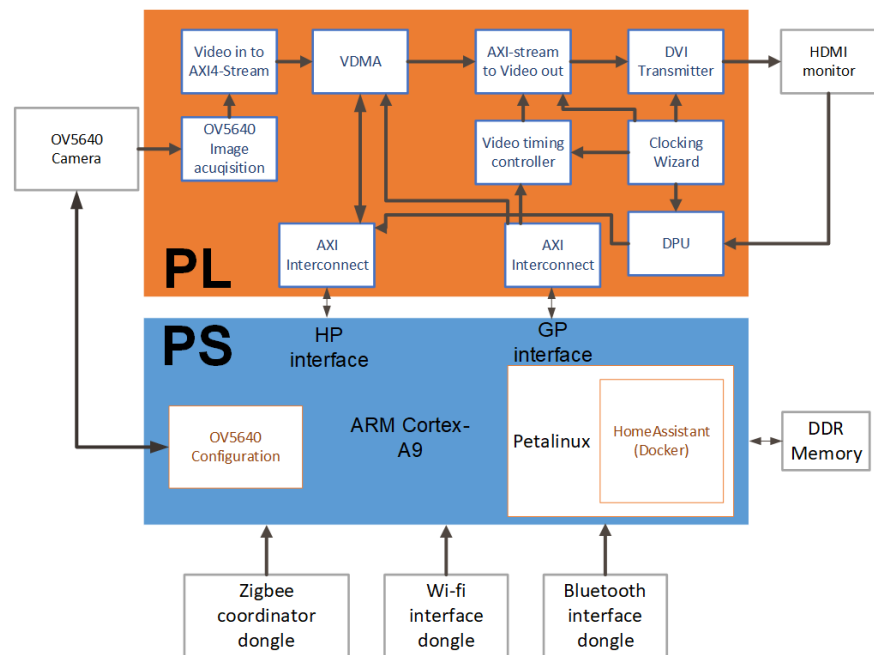


Figure 1. Block diagram of the core device

2.2. Video data acquisition

For capturing input image data OV5640 image sensor is used. It's capable of capturing images in QSX Video Graphic Array (QSXVGA) format (2592x1944 pixels) with 15 frames per second (FPS) or in VGA format (640x480) with 90 fps. It's connected to SoC via "camera module connection" slot by pins. OV5640 uses proprietary Serial camera control bus (SCCB) protocol which control most functions of image sensor, like image processing parameters or resolution, image output interface. The system utilizes an OV5640 CMOS sensor, which supports 5-megapixel output and is configured by default in RGB565 format. Raw image data is received via the MIPI interface - after deserialization, it is converted into a parallel RGB data stream and transmitted to "Video to AXI4-Stream" module.

2.3. Video data transformation

The "Video in to AXI4-Stream" IP module is used to convert parallel video signals from OV5640 image sensor into the "AXI4-Stream" data transfer protocol, ensuring compatibility with the Video Direct Memory Access (VDMA) IP module.

The collected image data need to be transferred for DDR frame buffer, for that “VDMA” module is used, which can be operated only with “AXI4-Stream” data format. VDMA module configured in dynamic genlock mode with a three frame buffers in “Dynamic-Master” mode for write channel and in “Dynamic-slave” in read channel in order to prevent the read channel and the write channel from accessing the same frame buffer at the same time. The AXI4-stream input data is converted into the AXI4 Memory Map format through the VDMA write channel, and finally written into the DDR memory. VDMA connects to the AXI_HP port through the AXI Interconnect IP core to efficiently access DDR3. The video or image data read by VDMA from DDR3 is transmitted to the AXI4-Stream to Video Out IP core.

The “AXI4-Stream to Video out” IP module is used to convert the AXI4-Stream format data into RGB565 format data under control of Video Timing Controller (VTC) IP and sends it to “DVI Transmitter” module input.

Video Timing Controller IP serves as a universal module for detecting and generating video stream timing signals, ensuring synchronization between video path components when working with the AXI4-Stream interface

“DVI Transmitter” module transform input RGB565 format data into Transition-minimized differential signaling (TMDS) format. The module accepts a 24-bit RGB video stream with a pixel clock and horizontal and vertical sync signals as input, and then performs TMDS encoding of the data and clock in accordance with the Digital Visual Interface specification. Encoding involves converting 8-bit color components into 10-bit TMDS sequences with 1080P (1920x1080 pixels) resolution that broadcasted on HDMI monitor.

2.4. Processing of video data

Deep learning processor unit (DPU) takes for input data broadcasted on HDMI monitor video stream for Human Activity Detection (HAR) task using CNN model for detection, labeling and logging such type of activity like for each householder: entering home, leaving home, start/stop watching TV, start/stop working on computer, eating on kitchen, etc. DPU IP is a configurable computation engine, that is responsible for CNN computation. DPU is capable to work with any CNN module: It’s can make convolution operations with maximum 16x16 kernel, deep with convolution, deconvolution and max poling operations. DPU can’t execute CNN operations by its own – commands and input images should be provided by Application processing unit (APU), which in this work is presented as a specially build Linux OS installed on SoC PS side. For working DPU requires two clock timings – s_axi_aclk for register configuration model and twice as fast dpu_2x_clk for computation.

Clocking wizard IP give control to configure clock circuit for user requirements. On input clocking wizard connected to 100MHz ZYNQ7 Processing system PL fabric clock, that using Mixed-Mode Clock Manager (MMCM) primitive generates

next output clocks: 300 MHz dpu_2x_clk and 150 MHz s_axi_aclk for DPU module; 75 MHz clock for VTC, “AXI4-Stream to Video Out” and pclk input of “DVI_Transmitter” module; 375 MHz clock for pclk_x5 input of “DVI_Transmitter” module.

3. Software design

3.1. Operating system

ZYNQ UltraScale+ MPSoc device is capable of running embedded lightweight Linux OS. This OS been specially build using Petalinux Software Development Kit (SDK). Linux OS filesystem is configured in ext4 format for enabling opportunity of booting OS from high memory capacity SD card. Petalinux SDK can implement different software libraries in Linux Root File system (RootFS). The next libraries and packages are chosen for implementing: DPU model – for enabling DPU capabilities inside embedded Linux; sudo – for ability to provide superuser rights to users and programs; Petalinux python modules – for ability to execute python scripts on embedded Linux; X11 package – for enabling windowing system and GUI visualization; OpenCV libraries – for executing CNN model related code; petalinux-networking-stack – for extending network stack available settings capability for enabling Ethernet and WI-FI interface configuration. Docker is established in the OS using method of creating a custom Yocto layer with including program code of Docker.

3.2. Deployment system

Docker software is a platform for application containerization that allows software, along with all its dependencies, libraries, and configurations, to be encapsulated into isolated environments called containers, which ensure consistent execution across any host system, regardless of its operating system. A Docker container contains all the components needed to run an application, which eliminates compatibility issues between development and production environments, ensures isolation from other programs on the host machine, and allows for efficient use of computing resources with minimal overhead. Above concludes that “Container” approach is a perfect way of realization a “Home Assistant” software ecosystem for devices with limited memory resources, like ZYNQ UltraScale+ MPSoc.

3.3. Smart home hub system

Home Assistant is an open-source platform for home automation. It is free software designed to centrally control and automate smart devices and home appliances, providing a unified control interface regardless of the device manufacturer. Unlike proprietary ecosystems, Home Assistant is installed locally

on hardware and does not require a connection to cloud services for its core functionality. Device with installed “Home assistant” software is called the “Hub”. in IoT network. Home assistant use WI-FI, Bluetooth and Zigbee wireless protocols for transferring data between hub and IoT devices (sensors, smart home appliance, actuators). Zigbee capabilities enabled using MQ Telemetry Transport (MQTT) protocol broker. Zigbee coordinator dongle CC2652P collects and sends information from Zigbee IoT devices to Hub. The platform uses integrations to connect devices from various manufacturers. Automations is enables through scripts and rules that are activated by triggers that monitor information from DPU generated log files and IoT system sensors. Result of automation execution is a process of performing action via actuators (open/close the relay) and smart home appliance (turning devices on/off, sending notifications, etc).

4. Experimental part

Next experiment is concluded for comparisons object detection time results between FPGA-based and CPU-based accelerators of CNN model. The experiment involves measuring the inference time required by the CNN model to process one image. Inference time is a time that CNN spend for object detection, labeling, and generating bounding boxes. Experimental input data consist of 80 classes in 5000 sample images.

YOLOv3 algorithm was selected as the model. The model been trained on COCO train Val 2017 dataset, which contains 123.827 images with 886.284 annotation instances for more than 80 different classes: humans, vehicles (cars, planes), animals (cats, dogs), sports equipment (balls, boards), kitchenware (forks, spoons), food (bananas, hotdogs,), furniture (bed, chair, etc.), electronics (computers, phones). Model version for FPGA-based accelerator been additionally modified. The framework of YOLOv3 model been transformed from darknet-53 to TensorFlow. Additional quantization process been made to bypass FPGA SoC memory restrictions. Final modification is a compilation of CNN model in readable for DPU binary representation.

For FPGA-based accelerator Xilinx Zynq 7000 SoC FPGA “XC7Z020-CLG400-2” been chosen, which hardware design is built according to the Figure 1 block diagram. It’s Processing System (PS) side is based on dual-core ARM Cortex-A9 CPU, Programmable Logic (PL) side is based on Artix 7 FPGA, and have 85.000 programmable logic cells, 53.200 LUT, 106.400 flip-flops, 4.9 Mb of block RAM and 220 DSP slices.

For CPU-based accelerator “Steam Deck” computer is used. It’s CPU is “AMD Zen 2 4c/8t” with power consumption 15 w, which capable of getting to 3.5GHz clock speed. The operation system for conducting CPU experiment is Ubuntu 22.04.5 desktop. Execution of YOLOv3 script is done in “AlexeyAB” version of DarkNet53

framework.

The results of the experiment were plotted in the graph shown in Figure 2. Experiment shows, that average inference time for CPU-based CNN YOLOv3 model is 2.277 seconds. Minimum inference time from 5000 samples is 2.281 seconds, maximum inference time is 3.406 seconds. CPU inference time graph has fluctuation spikes (11 in observed batch), that can be go up to 3.4 seconds. The average inference time for FPGA-based CNN YOLOv3 model is 0.77 seconds. Minimum inference time from 5000 samples is 0.763 seconds, maximum inference time is 0.797 seconds. FPGA graph is smooth, without spikes, with $\Delta \pm 0.02$.

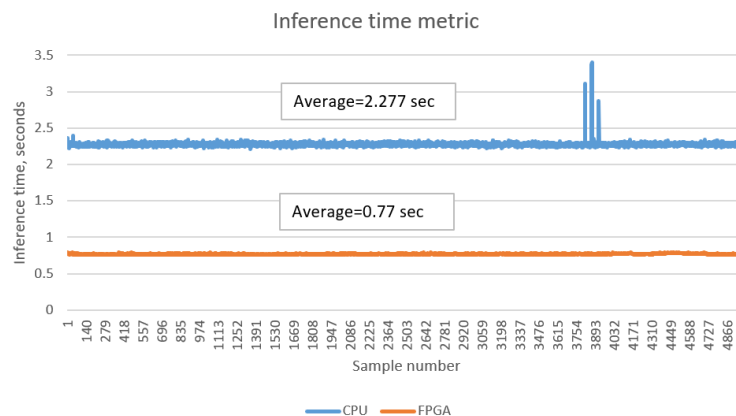


Figure 2. Graph of inference time experiment result

The Table 1 summarizes the differences in the characteristics of the two approaches of creating CNN accelerator for IoT smart home system. Based on the characteristics it's possible to conclude, that CNN FPGA accelerator is much faster than CPU-based equivalent.

Table 1. Characteristics of CNN accelerators

CNN type	Characteristics				
	Max inference time ,s	Min inference time, s	Avg inference time, s	Clock speed, MHz	Power consumption, W
CPU-	3.406	2.281	2.277	3500	15
FPGA	0.797	0.763	0.77	300	3.5

5. Conclusion

In this paper proposed concept for using FPGA-based computer vision in smart home IoT, that can be also implemented in different branches of IoT paradigm. The core design of system is proposed for implementation on Xilinx Zynq SoC FPGA. Suggested system is ecological because of minimal consumption of power; robust and reliable against cyberattacks because of its edge structure that not depends on cloud servers; versatile and easy to use because of using "Home assistant" software that can control IoT devices with different types of wireless protocols like WI-FI,

Zigbee and Bluetooth.

Experiment been concluded, which shows difference in CNN computation inference time between FPGA-based solution and CPU-based solution. The results of the experiment show that FPGA-based solution operates faster than CPU-based solutions, even though the CPU-based solutions have higher clock speeds and consume more power. That makes FPGA-based CNN accelerator to be perfect choice for solving computer vision related tasks in IoT networks.

References

- [1] Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, E., Ligett, K., Lyons, T., Manyika, J., Niebles, J.C., Shoham, Y., Wald, R., Walsh, T., Hamrah, A., Santarlaschi, L., Lotufo, B.L., Rome, A., Shi, A., Oak, S. (2025) The AI Index 2025 Annual Report. AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Stanford, CA, April 2025.
<https://doi.org/10.48550/arXiv.2504.07139>
- [2] Elouardi, S., Motii, A., Jouhari, M., Nasser Hassane Amadou, A. and Hedabou, M. (2024) A Survey on Hybrid-CNN and LLMs for Intrusion Detection Systems: Recent IoT Datasets. IEEE Access, 12, 180009-180033.
<https://doi.org/10.1109/ACCESS.2024.3506604>
- [3] Khan, H., Yuan, X., Qingge, L. and Roy, K. (2025) Violence Detection From Industrial Surveillance Videos Using Deep Learning. IEEE Access, 13, 15363-15375, 2025.
<https://doi.org/10.1109/ACCESS.2025.3531213>.
- [4] Aldossary, M., Alharbi, H.A. and Anwar Ul Hassan, C. (2024) Internet of Things (IoT)-Enabled Machine Learning Models for Efficient Monitoring of Smart Agriculture. IEEE Access, 12, 75718-75734.
<https://doi.org/10.1109/ACCESS.2024.3404651>
- [5] Gad MM, Gad W, Abdelkader T, Naik K. (2025) Personalized Smart Home Automation Using Machine Learning: Predicting User Activities. Sensors (Basel). 2025 Oct 2;25(19),6082.
<https://doi.org/10.3390/s25196082>
- [6] Wang, F., Zhang, M., Wang, X., Ma, X. and Liu, J. (2020) Deep Learning for Edge Computing Applications: A State-of-the-Art Survey. IEEE Access, 8, 58322-58336, 2020.
<https://doi.org/10.1109/ACCESS.2020.2982411>
- [7] Rosero-Montalvo, P.D., Tözün, P. and Hernandez, W. (2024) Optimized CNN Architectures Benchmarking in Hardware-Constrained Edge Devices in IoT Environments. IEEE Internet of Things Journal, 11, 20357-20366, 1 June1, 2024.
<https://doi.org/10.1109/JIOT.2024.3369607>