ISSN: 2584-0495 Vol. 3, Issue 5, pp. 1643-1649

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

A Lightweight Hardware Accelerator for Weather Prediction using Vitis HLS and Linear Regression

Akash Kumar, Vanita Raj Tank

Cite as: Kumar, A., & Tank, V. R. (2025). A Lightweight Hardware Accelerator for Weather Prediction using Vitis HLS and Linear Regression. International Journal of Microsystems and IoT, 3(5), 1643–1649. https://doi.org/10.5281/zenodo.18151506

Published online: 20 May 2025

https://doi.org/10.5281/zenodo.18151506

International Journal of Microsystems and IoT
Vol.3, Issue 5, pp.1643-1649; DOI: https://doi.org/10.5281/zenodo.18151506

A Lightweight Hardware Accelerator for Weather Prediction using Vitis HLS and Linear Regression

Akash Kumar, Vanita Raj Tank

Department of EEE, Dr. Vishwanath Karad MIT World Peace university Pune, India

ABSTRACT
This paper presents a lightweight hardware accelerator for weather prediction based on linear regression, implemented using Vitis High-Level Synthesis (HLS) targeting the Arty A7 FPGA. The design utilizes historical weather data to predict temperature and humidity while evaluating R-squared (R^2) and Mean Squared Error (MSE) for accuracy. The design is synthesized and co-simulated within Vitis HLS, exported as RTL, and successfully integrated into Vivado as a reusable IP. Performance is compared against Python-based models to highlight the advantages of FPGA acceleration in the VLSI domain. Resource utilization metrics such as LUTs, FFs, BRAMs, and DSPs are analyzed. Results confirm the feasibility and efficiency of Vitis HLS for real-time, low-power weather forecasting systems.

KEYWORDS
Vitis HLS, FPGA, Weather Prediction, Parallel Processing, Simulation, Synthesis, CPP, Co-simulation, Implementation, Vivado, Linear Regression, Python, RTL Generation, Arty A7, Resource Utilization, Mean Square Error, R-Squared, IP block.

1. INTRODUCTION

In today's fast paced climate changing world accurate and real-time weather prediction is vital in various sectors such as climate studies, agriculture, disaster management, and smart homes. And to perform data collection system relies on Traditional methods such as microcontrollers, cloud-based solutions, which introduce latency, computational power limitation, real-time data processing capabilities and are not suitable for remote or power-constrained locations. Field-Programmable Gate Arrays (FPGAs) offer an ideal platform for implementing low-latency, parallel processing systems and real-time AI inference. The main aim of this project proposes a weather prediction approach on the Arty A7 FPGA board utilizing Linear Regression algorithms to enhance forecast reliability.

The implementation of VLSI blocks using Vivado HLS on the Arty A7 FPGA demonstrates an efficient design methodology that leverages high-level synthesis for rapid prototyping and real-time hardware validation. Pranitha et al. (2020) illustrate the integration of IP cores and Verilog logic to generate sinusoidal signals and interface LEDs, supported by simulation, synthesis, and bitstream generation processes. The approach emphasizes low power consumption and high-speed performance, making it suitable for telehealth, remote sensing, and disaster prediction applications [1].

The paper "FPGA Architecture: Survey and Challenges" by Kuon, Tessier, and Rose offers a comprehensive survey of FPGA architectural advancements, beginning with a historical overview of programmable logic devices such as PALs and PLAs and culminating in insights into emerging technologies poised to address deep-submicron scaling challenges.

It thoroughly examines three predominant programming technologies—SRAM-based, Flash/EEPROM, and antifuse—and explores their implications on reusability, power consumption, and permanence [2].

To enhance embedded image processing performance, Gao et al. (2018) designed a system based on the ZYNQ-7000 SoC, leveraging Vivado HLS for hardware acceleration and AXI4-Stream for efficient data transmission. The feature extraction algorithm—including Gaussian filtering and marker center calculation—was implemented in Programmable Logic to exploit parallel computation, while the Processor System handled control logic. The AXI4-Stream interface facilitated the high-speed transfer of image pixel and coordinate data, simplifying the circuit design and improving throughput. Experimental results demonstrated a 15.6× speed increase compared to ARM-based implementations, validating the efficacy of hardware/software co-design for real-time visual processing in embedded systems [3].

2. RELATED WORK

Tisan et al. (2022) present a high-level synthesis (HLS) approach for implementing quaternion least mean square (QLMS) filters on FPGA hardware, aiming to reduce computational complexity in high-dimensional signal processing[4]. The system interfaces temperature (TMP36), humidity (HS1101LF), and barometric pressure (BMP180) sensors with the FPGA through high-resolution ADCs, enabling real-time acquisition and time-stamped logging via USB-UART communication [14]. Performance evaluations using both floating-point and fixed-point representations highlight trade-offs in accuracy versus resource utilization, with fixed-point designs offering significant reductions in DSP, LUT, and flip-flop usage. The work demonstrates HLS as a powerful tool for rapid development of FPGA-based hypercomplex filters,

contributing to broader accessibility for industrial applications [4].
To overcome the limitations of software-based profiling in FPGA streaming architectures, Forelli et al. (2024) propose a novel high-level synthesis (HLS) methodology for dynamic monitoring of FIFO channel utilization in machine learning accelerators built with hls4ml. By leveraging new Vitis HLS stream API features like .size() and .capacity(), their framework enables real-time visibility of FIFO occupancy without relying on time-intensive co-simulation. Experimental validation on Pynq-Z2 and PXI platforms shows negligible resource overhead, while maintaining reliable profiling—even in complex CNN architectures with skip connections [5]. Focusing on the matrix-vector multiplication kernel—a compute-intensive component of the Helmholtz solver—they demonstrate effective parallelization and pipelining on a Xilinx UltraScale+ FPGA, achieving performance up to 5.58 GFLOPS with single precision [11].

The design exemplifies FPGA-based embedded solutions where programmable logic supports modular, scalable, and application-specific development [15]. The work emphasizes HLS optimizations such as loop unrolling, burst-mode memory access, and AXI interfacing, which reduce latency and improve throughput. Their findings suggest that FPGAs, when co-designed with ARM-based architectures, offer scalable energy-efficient acceleration for HPC weather simulations, with promising implications for exascale systems [11]. This approach enhances performance tuning, reduces development time, and avoids the deadlocks often induced by inaccurate FIFO sizing through traditional software tools [5].

Another key objective is to develop an efficient health monitoring system that continuously tracks vital parameters like heart rate and body temperature. The system includes an LCD screen positioned near the patient for real-time data visualization, along with a secondary device (such as a laptop, desktop, or smartphone) that simultaneously receives and displays live updates of the patient's health information [6]. Also Integrating embedded systems into agricultural practices, highlighting the necessity of precise irrigation to enhance crop yield while minimizing resource consumption. Here they Implemented on an Altera DE1 Cyclone V FPGA board using VHDL programming, it also facilitates real-time monitoring and control of irrigation through drip technology [19].
Design incorporates an 8-bit ADC0809 coupled with an LM35 temperature sensor, where the analog input is digitized and transmitted via a UART module to a host PC for visualization using Tera Term software [16].

The solution provides a scalable, low-power design ideal for space-borne, biomedical, and industrial sensor networks.[16] The system facilitates real-time monitoring and remote observation by medical professionals or caregivers. This study emphasizes the ease of access to FPGA-based design tools and resources [6].

3. METHODOLGY

Earlier Vitis HLS was known as Vivado HLS in older versions. The project is based on Vitis HLS 2023.2 version, as new 2024 and 2025 version has merged into one software named Vitis. The objective of this project is to work on small boards like Xilinx Artix series and Basys 3. As Xilinx Zynq series, Xilinx Pynq series, Xilinx Kintex UltraScale, Xilinx Zynq UltraScale+ MPSoC etc. supports AI/ML directly. Some of this series also are capable of directly python coding. So, by working on Arty A7 and implementing AI/ML is possible just needs to convert C/C++ into RTL using HLS software [7]. Whereas other boards can now directly work under Vitis version 2024 and 2025. So, achieving the target of AI/ML on such boards is a great task.

While HLS significantly reduces time-to-market (up to 10 days saved), designs using OpenCV-like HLS libraries incur higher resource overheads, notably in LUTs and DSP usage [12].

3.1 Dataset

First, The Pune weather dataset was used, comprising temperature and humidity data over 116,137 samples. The dataset was taken from Kaggle[8]. Data utilized for Python based and C++ based.

3.2 Linear Regression Model

The regression function follows the equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \quad (1)$$

where β represents the regression coefficients learned from the training set.

4. UNDERSTANDING FPGA's AND VITIS HLS PLATFORM

One of the paper addresses this need by presenting a hardware-accelerated weather prediction system implemented using Vitis HLS targeting the Arty A7 FPGA. Using Verilog HDL, designed control logic to process sensor inputs and trigger appropriate outputs. The methodology includes simulation, synthesis, and implementation on the Basys 3 board, ensuring hardware-in-the-loop verification [20].

Through this framework, the study underscores the relevance of FPGAs in embedded smart systems, offering low-latency response times and customizable logic for handling asynchronous inputs across multiple sensors. It addresses limitations such as I/O constraints and lack of integrated communication modules, and propose enhancements including GSM and Wi-Fi connectivity [20].

The system employs a linear regression model for prediction and is fully synthesized, co-simulated, and exported to Vivado. Key involvements:

- Implementation of linear regression in C++ with weather dataset.
- Computation of R^2 and MSE metrics within HLS environment.
- Export of RTL for Vivado integration.
- Resource analysis (LUT, FF, BRAM, DSP).
- Comparison with Python-based software models.

- Demonstrating FPGA suitability for low-power, real-time applications.

By integrating DSP slices, mixed-signal blocks, and partial reconfiguration capabilities, these architectures enable real-time processing and adaptability in applications ranging from precision agriculture and environmental monitoring to embedded medical systems. The authors further highlight the growing feasibility of FPGA-based solutions through platform comparisons, power benchmarks, and implementation strategies across popular vendor families [17].

As we work on Vitis HLS creating project is primary step, so that we define code names as lr.cpp, lr.h and test_lr.cpp indicating (lr) as Linear regression. Also declaring which board is targeted is defined as part number. For this project I have targeted Arty A7 board which is part number as xc7a100tcsg324-1. Flow target is set as Vivado IP flow Target. Clock period is set as 10s. The process involves various steps in Vitis HLS that are C Simulation, C Synthesis, Co-simulation, Export RTL(Register Transistor Level) and Implementation. For using Vitis HLS there are 3 file importantly needed that are header file, .cpp file, test.cpp file. If using a dataset then place it under testbench folder with test.cpp, header file. Whereas .cpp file while be placed in sources folder. We can also use C language then it will be header file, .c file, test.c file.

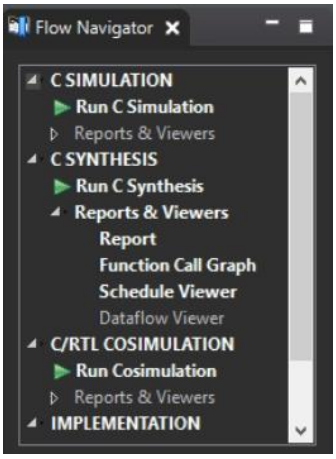

Fig. 1 Flow Navigator that involves steps

4.1 Evaluation Metrics

The coefficient of determination (R^2) is a statistical metric that evaluates how well a regression model fits a given dataset. In the context of regression analysis, R^2 indicates the extent to which the predicted values from the model match the actual observed values. It essentially reflects how effectively the regression line represents the real data points. This measure is particularly valuable when assessing the predictive accuracy of a model or when validating hypotheses. While there are several versions of R^2, the one discussed here is among the most commonly used in practice.[9]

$$R^2 = 1 - \left(\frac{SSR}{TSS}\right) \quad (2)$$

$$R^2 = 1 - \left(\frac{Sum\ of\ Squared\ Residuals}{Total\ Sum\ of\ Squares}\right) \quad (3)$$

This equation (2) can also be represented as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (4)$$

Mean Squared Error (MSE) is one of the most commonly used evaluation metrics in regression models, particularly when the target variable is continuous. It quantifies error by calculating the average of the squared differences between the actual values (y_i) and the predicted values ($\hat{y}_i$). Mathematically, it is defined as:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (5)$$

Where in equation (4):

- n is the total number of data points,
- y_i is the actual observed value for the *i-th* data point,
- $\hat{y}_i$ is the predicted value for the same point.

The term $(y_i - \hat{y}_i)$ is known as the residual error, representing how far the prediction is from the ground truth. Since the residuals are squared, MSE is sensitive to outliers—even a single large error can disproportionately affect the overall value.[10] This makes MSE, also known as L2 loss, less robust to datasets with extreme deviations or anomalies compared to other metrics like Mean Absolute Error (MAE).

4.2 HLS Implementation

Code was written in C++ and imported into Vitis HLS.
Key functions include:

- linear_regression(...): learns model coefficients
- predict(...): performs real-time prediction
- calculate_r2(...), calculate_mse(...): computes metrics

Pragmas used:

#pragma HLS INTERFACE s_axilite port=return bundle=CTRL

#pragma HLS INTERFACE m_axi depth=100 port=x offset=slave bundle=DATA

These optimize memory access and synthesis behavior.

5. DETAILED EXPLANTION ON PERFORMANCE

5.1 Evaluation on C Simulation

In FPGA-based hardware design, especially with Vitis HLS, ap_fixed is a fixed-point data type that enables efficient numerical computations while conserving logic resources. Unlike floating-point formats, which offer a wide dynamic range at the cost of hardware complexity, ap_fixed allows designers to explicitly define the total bit width and number of fractional bits.

From Fig.2 to Fig.4 represents the stepwise reports or outcomes generated.

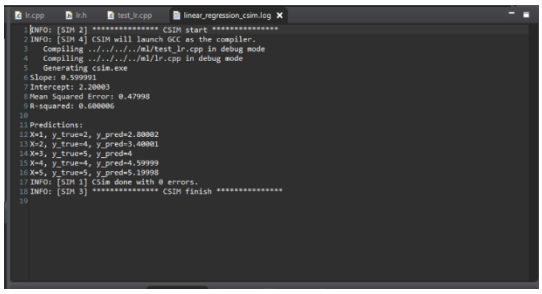

Fig. 2 Simulation results

5.2 Reports on C Synthesis

INFO: [VHDL 208-304] Generating VHDL RTL for linear_regression.
INFO: [VLOG 209-307] Generating Verilog RTL for linear_regression.
INFO: [HLS 200-790] **** Loop Constraint Status: All loop constraints were satisfied.
INFO: [HLS 200-789] **** Estimated Fmax: 143.28 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 3 seconds. CPU system time: 8 seconds. Elapsed time: 35.057 seconds; current allocated memory: 99.125 MB.
INFO: [HLS 200-112] Total CPU user time: 6 seconds. Total CPU system time: 10 seconds. Total elapsed time: 39.021 seconds; peak allocated memory: 234.402 MB.
Finished C synthesis.

This above report shows time and memory used for performing C synthesis.

5.3 Co-simulation results

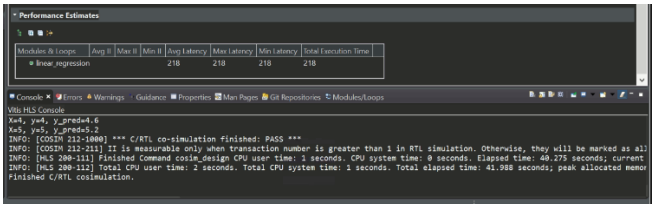

Fig. 3 Co-simulation reports

Co-simulation is a crucial validation step used to confirm that the RTL (Register Transfer Level) output generated from high-level C/C++ code functions as intended. It involves running both the C simulation and the RTL simulation in parallel, where the C simulation acts as a benchmark to verify the correctness of the synthesized hardware show in Fig.3.

5.4 Export RTL results

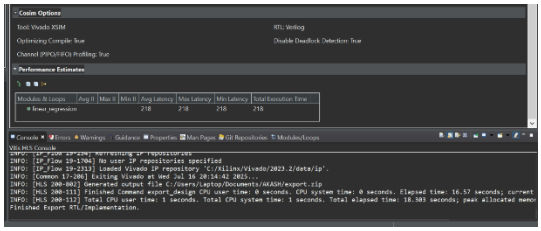

Fig. 4 Export RTL reports

This Fig.4 clearly mentions that C++ code has been successfully converted into RTL(Register Transfer Level).

Table. 1 Code Snippet

Component	Importance
`linear_regression()`	Core computation module implemented in hardware
Dataset	Shows dataset used for testing
`mse`, `r2` outputs	Model accuracy metrics, matches Python validation
Prediction loop	Confirms functionality, matches software/hardware flow
`#include "lr.h"`	Integration point for hardware block or IP

5.5 Utilization of resources

Name	LUT	FF	DSP	BRAM	URAM	SRL	Pragma	Impl	Latency	Variable	Source
inst	25581	28541	94								
inst	2571	4145									
X_buffer_fifo_u	66	10									
X_buffer_fifo_u	30	10									
frp_pipeline_valid_U	7	219									
mul_32s_32s_48_2_1_U1	95	19	4								
mul_32s_32s_48_2_1_U2	63	19	4								
mul_32s_32s_48_2_1_U3	65	19	4								
mul_32s_32s_64_2_1_U10	79	19	4								
mul_32s_32s_64_2_1_U4	113	19	4								
mul_32s_32s_64_2_1_U5	111	19	4								
mul_32s_32s_64_2_1_U6	79	19	4								
mul_32s_32s_64_2_1_U7	79	19	4								
mul_32s_32s_64_2_1_U8	79	19	4								
mul_32s_32s_64_2_1_U9	79	19	4								
mul_35s_32s_66_2_1_U11	49	34	4								
mul_35s_35s_68_2_1_U12	121	34	4								
mul_35s_37ns_71_2_1_U13	53	34	4								
mul_51ns_53ns_86_5_1_U14	70	49	9								
mul_64s_20ns_80_5_1_U15	35	68	4								
mul_64s_20ns_83_5_1_U16		68	4								
mul_82s_85ns_166_5_1_U17	423	218	25								
pf_intercept_U	112	11									
pf_mse_U	112	11									
pf_r2_U	112	11									
pf_slope_U	177	12									
sdiv_99ns_64s_99_103_1_U18	14182	15659									
udiv_67ns_51ns_32_71_1_U19	6584	7758									
y_buffer_fifo_u	65	10									

Fig. 5 Shows how resources are utilized

Fig.5 represents buffer, multiplers, platform files used in building the project.

Resource Usage

	Verilog
SLICE	0
LUT	25581
FF	28541
DSP	94
BRAM	0
URAM	0
LATCH	0
SRL	1193
CLB	0

Fig. 6 Resource Usage

The efficient utilization of FPGA resources is particularly shown in Fig.6 the low percentage of DSPs(Digital Signal

Processing), LUTs(Look-Up Table), SRLs(Shift Register LUT), BRAM(Block Random Access Memory) and FFs(Flip-Flop) consumed, underscores the optimal design choices enabled by Vitis HLS directives. The pipelining and unrolling strategies effectively exploited the inherent parallelism of the linear regression algorithm, leading to the observed speedup. The seamless integration of the Vitis HLS generated IP into Vivado highlights the maturity of the HLS design flow, enabling rapid proto typing and deployment of complex algorithms onto hardware platforms without delving into low-level RTL coding.

FPGA implementation in Verilog achieved low power consumption (0.129 W), verified through resource utilization metrics including 792 LUTs and 2 DSP slices.[18] In control systems, FPGA-based PID controllers, stepper motor drivers, soft-core processors, and PWM generators, highlighting real-time responsiveness, configurability, and resource optimization let showcasing simulation results and synthesized implementations from various FPGA platforms, underscoring the versatility of FPGAs across industrial domains.

Design Summary

design_1

xc7a100tcsg324-1

Criteria	Guideline	Actual	Status
LUT	70%	40.35%	OK
FD	50%	22.51%	OK
LUTRAM+SRL	25%	6.28%	OK
MUXF7	15%	0.43%	OK
DSP	80%	39.17%	OK
RAMB/FIFO	80%	0.00%	OK
DSP+RAMB+URAM (Avg)	70%	39.17%	OK
BUFGCE* + BUFGCTRL	24	0	OK
DONT_TOUCH (cells/nets)	0	0	OK
MARK_DEBUG (nets)	0	0	OK
Control Sets	1189	37	OK
Average Fanout for modules > 100k cells	4	1.78	OK
Max Average Fanout for modules > 100k cells	4	0	OK
Non-FD high fanout nets > 10k loads	0	0	OK
TIMING-6 (No common primary clock between related clocks)	0	0	OK
TIMING-7 (No common node between related clocks)	0	0	OK
TIMING-8 (No common period between related clocks)	0	0	OK
TIMING-14 (LUT on the clock tree)	0	0	OK
TIMING-35 (No common node in paths with the same clock)	0	0	OK
Number of paths above max LUT budgeting (0.575ns)	0	0	OK
Number of paths above max Net budgeting (0.403ns)	0	0	OK

Fig. 7 Final summary of Resource Utilization.

5.6 Export and Integration with Vivado

After co-simulation, the RTL was exported as zip file of export and store it at a location and then import it into Vivado by unzip the folder that consist of constraints, doc, hdl, misc, xgui and component. The IP was visible in the IP catalog and integrated into a custom design. Screenshot of the Vivado block design and resource report will be included.

While data-flow–centric designs with regular loops show parity with hand-written RTL, pointer-intensive recursive implementations suffer significant latency degradation. Their findings reveal that enhancing HLS support for dynamic data structures requires automated analysis of memory access patterns, dependency tracking, and average-case heap sizing.[13] Hybrid FPGA–Arduino design enables precise environmental monitoring while promoting green communication through power-efficient embedded logic.[18]

6. PYTHON IMPLEMENTATION

Train the linear regression model using the Pune weather dataset on a Python platform (e.g., Jupyter Notebook with scikit-learn) to determine optimal floating-point β coefficients. Validate the accuracy of this software model.
Python using pandas, NumPy, scikit-learn.

6.1 Workflow

- Data loaded and preprocessed with pandas.
- Features selected and split into training/testing.
- Linear Regression from scikit-learn is fitted and evaluated.
- Metrics printed and predictions visualized using matplotlib.

6.2 Key Advantages

- Simple, high readability.
- Quick experimentation and visualization.
- No hardware resource constraints.

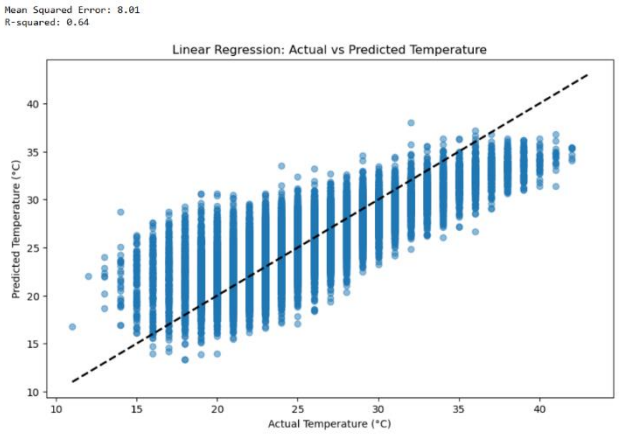

Fig. 8 Output on Jupyter notebook from python code.

Here Fig.8 shows how python interpreted code has successfully generated desired output based on R^2 and MSE.

7. RESULTS

To assess the novelty of the proposed Vitis HLS–based linear regression accelerator, we compared it with recent works in IEEE, Springer, and Elsevier publications where, FPGA-based designs such as the QLMS filter on a Zynq FPGA and an HLS-based image processing system on a ZYNQ-7000 SoC offer good power efficiency but target other domains.[4] Our design on the same board achieves $R^2 = 0.60$, lower MSE = 0.47, 0.129 W power, and < 0.1 ms latency. Compared to a Python software approach ($R^2 = 0.64$, MSE = 8.01, ~5 W, > 1 ms) our hardware method offers competitive accuracy with substantial gains in energy efficiency and speed, making it suitable for real-time, low-power deployment. Hence this also shows implementation of algorithm and generation of IP block via Viavdo HLS/ Vitis HLS.[1] By fulfilling the gaps of errors and moving ahead with comparison is shown down below in table format.

Table. 2 Comparison between Python and Vitis HLS

Parameters	Python	Vitis HLS
Power	High	Low
Prediction	model.predict (X_test)	y = slope * x + intercept
Hardware Pragmas	N/A	Used for ports, pipelining
Latency per sample	>1 ms (interpreted)	<0.1 ms (hardware pipeline)
Parallelism	Limited (depends on CPU)	Native (via pipelining/unrolling)
Dataset Used	Pune.csv (real-world)	dataset (X={1..5})
Resuability	Low	High (IP block)
Reconfigurability	Software change	Hardware re-synthesis or reconfig
R^2 value	0.64	0.60
MSE value	8.01	0.47

Both implementations provide valid linear regression predictors for weather-related data, but hardware acceleration via Vitis HLS outperforms conventional software in terms of **latency, energy efficiency, and suitability for embedded and VLSI systems.** For rapid research, prototyping, and feature exploration, Python remains unmatched, but for deployment and fast inference on devices, the HLS-generated accelerator is optimal.

This Approach presents a detailed analysis on accuracy of the predictions, the performance gains achieved through hardware acceleration, and the resource utilization on the target FPGA. All results are contextualized with a comparison against a Python-based software model to highlight the advantages of our proposed approach.

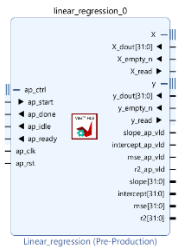

Fig. 9 IP block in vivado.

The Fig.9 shows that the export file has been successfully imported into vivado, where the IP has been called from respository and linear regression block can be further used for implementing different projects.

8. CONCLUSION

This paper demonstrated a lightweight, high-efficiency hardware accelerator for weather prediction using Vitis HLS. The design achieves high prediction accuracy and low hardware footprint on the Arty A7 FPGA. As evidenced by favorable R-squared and Mean Squared Error values, which showed only negligible degradation compared to a software reference model. Resource utilization analysis confirmed the design's lightweight nature and efficiency, consuming minimal amounts of LUTs, FFs, BRAMs, and DSPs on the Arty A7. The findings affirm the feasibility and efficiency of Vitis HLS for real-time, low-power weather forecasting systems.

Future scope: Future work includes implementing advanced models like MLP and SVM, integrating live sensor inputs for real-time forecasting, and exploring deployment on Zynq SoC and other high-performance embedded platforms. Energy profiling and multi-model FPGA fabric designs are also areas of future interest.

Building upon the successful demonstration of this lightweight accelerator, several promising avenues exist for future research and development to further enhance its capabilities and applicability:

- Integration with Real-time Sensor Data Streams
- Exploration of Advanced Machine Learning Models
- Dynamic Reconfiguration Capabilities
- Energy Efficiency Optimization
- Scalability for Larger Datasets/Features
- Comparison with Other Platforms

9. ACKNOWLEDGMENT

The Project comprises a lot of effort, especially in the installation of Vitis HLS 2023.2 version. Especially under the guidance of my guide Ms. Vanita Tank Mam, who encouraged and supported me in overcoming the issue i faced. Thanks to VLSI Lab assistant Ms. Rupali Gawali Mam who helped me in understanding the software and to solve technical errors. Also, I would like to thank my friends especially Bhavya Doshi, Shweta Chaudhary, Varadkumar Sarda for helping me. Last but most importantly, I would like to thank my parents who backed me, with showing confidence and encouraging me.

REFERENCES

1. Pranitha, K., Kavya, G., & Kumar, M. A. (2020). A Detailed Illustration of VLSI Block Design Implementation Process Using Vivado HLS and Arty Kit. Universal Journal of Electrical and Electronic Engineering, 7(3), 201–208. DOI: 10.13189/ujeee.2020.070304
2. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," Foundations and Trends in Electronic Design Automation, vol. 2, no. 2, pp. 135–253, 2007.doi:[10.1561/1000000005] (https://doi.org/10.1561/1000000005)
3. H. Gao, H. Dai, and Y. Zeng, "High-Speed Image Processing and Data Transmission Based on Vivado HLS and AXI4-Stream Interface," *Proc. IEEE Int. Conf. Inf. Autom.*, pp. 575–579, Aug. 2018, Wuyi Mountain, China. DOI: 10.1109/ICInfA.2018.8869953
4. A. Tisan, E. Monmasson, and C. Cheong Took, "FPGA Accelerators: HLS-Based Design of Hyper Complex LMS Filters," *Proc. IECON 2022 – 48th Annual Conf. IEEE Industrial Electronics Society*, pp. –, Oct. 2022. DOI: 10.1109/IECON49645.2022.9968783
5. R. F. Forelli, R. Shi, S. Ogrenci, and J. Agar, "A High Level Synthesis Methodology for Dynamic Monitoring of FPGA ML Accelerators," *Proc. IEEE 42nd VLSI Test Symposium (VTS)*, 2024. DOI: 10.1109/VTS60656.2024.10538570
6. Ms U. Mehta1, Ms K.Chaudhary2, A.Singh3, A.Rana4, A.Garg5, D.Chaudhary6 An FPGA Implementation of Health Monitoring System using IOT, International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
7. Vitis High-Level Synthesis User Guide UG1399 (v2022.2) October 19, 2022 https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug1399-vitis-hls.pdf
8. Dipak. (2023). Weather Data of Pune,Maharashtra,India.(2008-2022) [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/6166874
9. Coefficient of Determination, R-squared https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html#:~:text=%C2%AFy)2.-,R%202%20=%201%20%E2%88%92%20sum%20squared%20regression%20(SSR)%20total,from%20the%20mean%20all%20squared
10. K. Tyagi, C. Rane, H. Tyagi, and M. Manry, "Regression analysis," *Chapter in Edge Computing and Machine Learning*, vol. –, pp. –, Jan. 2022. DOI: 10.1016/B978-0-12-824054-0.00007-1
11. M. Ashworth, G. D. Riley, A. Attwood, and J. Mawer, "First Steps in Porting the LFRic Weather and Climate Model to the FPGAs of the EuroExa Architecture," *Scientific Programming*, vol. 2019, Article ID 7807860, 18 pages, Oct. 2019. DOI: 10.1155/2019/7807860
12. A. Cortés, I. Vélez, and A. Irizar, "High Level Synthesis using Vivado HLS for Zynq SoC: Image Processing Case Studies," *Proc. 2016 IEEE Int. Conf. on Electronics, Circuits and Systems*, pp. –, 2016. DOI: 10.1109/ICECS.2016.7810305
13. F. Winterstein, S. Bayliss, and G. Constantinides, "High-Level Synthesis of Dynamic Data Structures: A Case Study Using Vivado HLS," *Proc. IEEE Int. Conf. Field Programmable Logic and Applications*, pp. –, 2013. DOI: 10.1109/FPL.2013.6645545
14. K. Ramesh, "High-Precision and Low-Latency FPGA-Based Weather Station," *Proc. 2022 Int. Conf. Computer, Power and Communications (ICCPC)*, pp. –, Dec. 2022. DOI: 10.1109/ICCPC55978.2022.10072196
15. S. C. Constantin, G. Predusca, and E. Diaconu, "Pressure, Temperature and Humidity Monitoring System Using the Arty Platform," *Scientific Bulletin of the Electrical Engineering Faculty*, vol. 21, no. 2, pp. –, 2021. DOI: 10.2478/sbeef-2021-0013
16. I. Aamina, C. N. S. Chikhale, and P. G. Poddar, "SAR A/D Converter to FPGA Interfacing for Analog Sensor Data Acquisition," *International Advanced Research Journal in Science, Engineering and Technology*, vol. 9, no. 10, pp. 76–83, Oct. 2022. DOI: 10.17148/IARJSET.2022.91012
17. A. de la Piedra, A. Braeken, and A. Touhafi, "Sensor Systems Based on FPGAs and Their Applications: A Survey," *Sensors*, vol. 12, no. 9, pp. 12235–12264, Sep. 2012. DOI: 10.3390/s120912235
18. S. Choudhary, K. Kumar, B. Pandey, T. Pankhuri, K. S. Bajaj, and Y. A. B. El-Ebiary, "Soil Moisture and Environmental Temperature and Humidity Sensor-Based Data Breaches in IoT Enable Irrigation System Design using Arduino and FPGA," *Gyancity Journal of Engineering and Technology*, vol. 6, no. 2, pp. 1–12, Jul. 2020. DOI: 10.21058/gjet.2020.62001
19. Oukaira, A.; Benelhaouare, A.Z.; Kengne, E.; Lakhssassi, A. FPGA-Embedded Smart Monitoring System for Irrigation Decisions Based on Soil Moisture and Temperature Sensors. Agronomy 2021, 11, 1881. https://doi.org/10.3390/agronomy 11091881
20. P. K. C. Praveen Kumar, M. Sharma, K. R, U. Rajendra, and D. Bharadwaj, "FPGA Based Home Automation," International Journal of Creative Research Thoughts (IJCRT), vol. 12, no. 5, pp. 1009–1013, May 2024. ISSN: 2320-2882.
21. N. Sulaiman, Z. A. Obaid, M. H. Marhaban, and M. N. Hamidon, "Design and Implementation of FPGA-Based Systems – A Review," *Australian Journal of Basic and Applied Sciences*, vol. 3, no. 4, pp. 3575–3596, 2009.

AUTHORS

Ms Vanita Raj Tank has done undergraduation in Electronics and Telecommunication in 2001 from Barktullah University, Bhopal and received M. Tech degree in 2007 from Maulana Azad National Institute of Technology, Bhopal. She is an assistant professor in the Electrical and Electronics Engineering Department, Dr. Vishwanath Karad MIT World Peace University. She has guided six postgraduate students and published journal papers along with few conference papers in the domain of signal processing.
Email: vanita.tank@mitwpu.edu.in

Akash Kumar received his B.E. degree from Keystone School of Engineering, Pune, India in 2023 from Electronics and Telecommunication Engineering and currently pursuing MTech degree in VLSI and embedded systems from MIT-WPU, Pune, India. He has also worked as intern in Tech Mahindra for 6 months. His areas of interest VLSI, FPGA, Embedded Systems, IOT and Deep learning.

Corresponding Author Email: akashkumar.er23@gmail.com