Close this dialog

Deepak Bhaskar Acharya;Karthigeyan Kuppan;B. Divya

Search by Content Type

Search within Publication

Search by Content Type

Search within Publication

Aims & Scope

IEEE *Access*® is a multidisciplinary, open access (OA), applications-oriented, all-electronic archival journal that continuously presents the results of original research or development across all IEEE fields of interest.

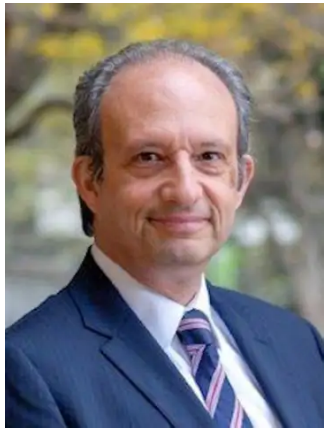
IEEE *Access* will publish articles that are of high interest to readers, original, technically correct, and clearly presented. Supported by article processing charges (APCs), its hallmarks are a rapid peer review and publication process with open access to all readers. Unlike IEEE's traditional Transactions or Journals, reviews are "binary", in that reviewers will either Accept or Reject an article in the form it is submitted in order to achieve rapid turnaround.

Open Access Article Processing Charge

- Submit Manuscript
- Submission Guidelines
- Become a Reviewer
- Open Access Publishing Options

Editorial Board

The IEEE *Access* Editorial Board members are IEEE Life Fellows, IEEE Fellows, or IEEE Senior Members who are recognized as technical experts in their fields. You can learn more about their responsibilities by visiting our Editorial Leadership (/editorial-leadership/) page. Below you will find a complete list of our board members. You can click on the bio link to learn more about each member.



DEREK ABBOTT

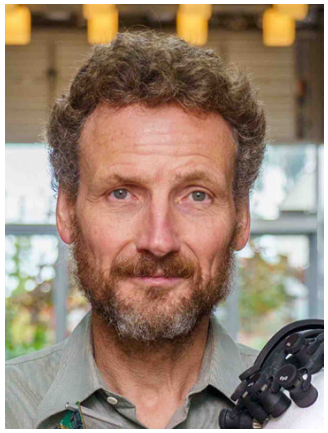
The University of Adelaide, Adelaide, SA, Australia

View Bio → (https://ieeeaccess.ieee.org/editorial-team/derek-abbott/)

KAREN BUTLER-PURRY

Texas A&M University, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/butler-purry-karen/)



GERT CAUWENBERGHS

University of California San Diego, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/cauwenberghs-gert/)

ZHIPENG CAI

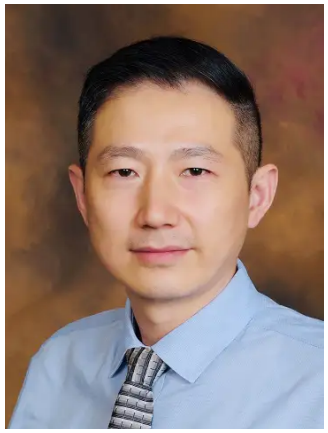
Georgia State University, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/zhipeng-cai/)

PIN-YU CHEN

IBM Thomas J. Watson Research Center

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/chen-pin-yu/)



YIXIN CHEN

Washington University, St. Louis, MO, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/yixin-chen/)

J.-C. CHIAO

Southern Methodist University, Dallas, TX, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/j-c-chiao/)

KERSTIN DAUTENHAHN

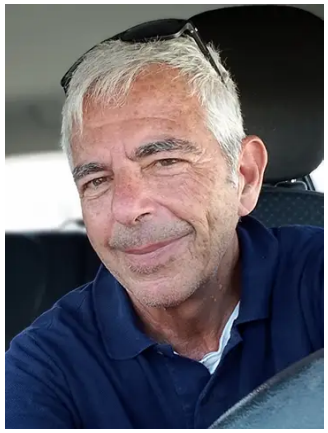
University of Waterloo, Ontario, Canada

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/dautenhahn-kerstin/)

AYMAN S. EL-BAZ

University of Louisville/Bioengineering Department, Louisville, KY, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/ayman-s-el-baz/)



LUIGI FORTUNA

University of Catania, Italy

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/luigi-fortuna/)

L. JAY GUO

The University of Michigan, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/guo-l-jay/)

HOAY B. GOOI

Nanyang Technological University, Singapore

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/hoay-b-gooi/)

PEDRO INACIO

University of Beira Interior (UBI), Portugal

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/inacio-pedro/)

ATIF IQBAL

Qatar University, Doha, Qatar

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/atif-iqbal/)

BAHRAM JAVIDI

University of Connecticut, Storrs, CT USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/bahram-javidi/)

YUMI IWASHITA

NASA-JPL California Institute of Technology, Pasadena, CA, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/yumi-iwashita/)

CECILE JUNG-KUBIAK

Jet Propulsion Laboratory, Pasadena, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/cecile-jung-kubiak/)

HULYA KIRKICI

University of South Alabama, Alabama, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/hulya-kirkici/)

GITTA KUTYNIOK

Ludwig-Maximilians-Universität München, Munich, Germany

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/gitta-kutyniok/)

AGNIESZKA KONCZYKOWSKA

ADesign

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/agnieszka-konczykowska/)

GERMANO LAMBERT-TORRES

Gnarus Institute, Itajuba, Brazil

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/germano-lambert-torres/)

XIAOLI LI

Singapore University of Technology and Design (SUTD)

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/xiaoli-li/)

NIGEL LOVELL

UNSW Sydney, Kensington, Australia

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/nigel-lovell/)

XIULING LI

University of Texas, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/li-xiuling/)

IMRAN MEHDI

Jet Propulsion Laboratory, California, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/imran-mehdi/)

MICHAL MROZOWSKI

Gdansk University of Technology, Gdansk, Poland

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/michal-mrozowski/)

TORU NAMERIKAWA

Keio University, Yokohama, Japan

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/toru-namerikawa/)

YI LU MURPHEY

University of Michigan-Dearborn, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/murphey-yi-lu/)

MICHELE NAPPI

University of Salerno, Fisciano, Italy

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/michele-nappi/)

DALMA NOVAK

Octane Wireless, Hanover, Maryland, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/dalma-novak/)

SUSANTO RAHARDJA

Singapore Institute of Technology/Infocomm Technology Cluster, Singapore, Singapore

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/susanto-rahardja/)

ARNOLD (NEVILLE) PEARS

KTH Royal Institute of Technology, Stockholm, Sweden

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/arnold-neville-pears/)

GIANLUCA SETTI

King Abdullah University of Science and Technology (KAUST), KSA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/gianluca-setti/)

GEORGIA TOURASSI

Oak Ridge National Laboratory (ORNL), Tennessee, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/georgia-tourassi/)

HENGYONG YU

University of Massachusetts Lowell, Lowell, MA, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/hengyong-yu/)

KEITH WEAR

U.S. Food and Drug Administration, Silver Spring, MD, USA

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/keith-wear/)

CHIK PATRICK YUE

Hong Kong University of Science and Technology, Hong Kong

View Bio →
(https://ieeeaccess.ieee.org/editorial-team/chik-patrick-yue/)

w.facebook.com/ieeeaccess) ’x.com/ieeeaccess) din.com/company/ieee-access) pp/profile/ieeeaccess bsky.social) org/app/company/6/IEEE-Access/activities)

Submit an Article ↗
(https://ieee.atypon.com/journal/ieee-access)

View Published Articles
(https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639)

(https://www.ieee.org/)

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 October 2024, accepted 28 October 2024, date of publication 4 November 2024, date of current version 15 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3491632

RESEARCH ARTICLE

Development of a Model for Prediction of IoT Processor Power Utilization Using Instruction Dissection-Based Technique

DIBAL P. YUSUF[1], HASHIM E. AHMED[2], (Member, IEEE), ELIZABETH N. ONWUKA[3], AND ZUBAIR SULEIMAN[3]

[1]Computer Engineering Department, University of Maiduguri, Maiduguri 600282, Nigeria
[2]Computer Engineering Department, College of Computer Science, King Khalid University, Abha 61421, Saudi Arabia
[3]Telecommunication Engineering Department, Federal University of Technology Minna, Minna 920101, Nigeria

Corresponding author: Dibal P. Yusuf (dibalpeter@unimaid.edu.ng)

This work was supported in part by the Deanship of Research and Graduate Studies at King Khalid University through the Large Research Project under Grant RGP2/291/45, and in part by the Tertiary Education Trust Fund (TETFUND) National Research Fund under Grant TETFUND/RD&D/NRF/STI/56/Vol.1.

ABSTRACT The global drive towards a net-zero world and commitments from advanced countries towards a significant cut of greenhouse gases by 2030 are objectives that have compelled a new technological paradigm that must align with new global initiatives towards meeting agreed objectives towards the net-zero world. The Internet-of-Things (IoT) is a technology that has shown strong viability in assisting nations and corporations in meeting global commitments to the net-zero goal. To be effective, the operation of IoT devices must meet very tight constraints; consequently, the design of the electronics, and applications that run on IoT devices must be optimal especially in power utilization considerations. It is in this regard that this paper presents the design of a power prediction model for a Cortex M processor. The design was achieved through a unique methodology in which an instruction dissector played a central role in determining the exact instructions that were executed, rather than using a generalized set of instructions from an instruction grouping. Multivariate Adaptive Regression Splines (MARS) was used as the machine learning technique for deriving the prediction models. A key feature of using MARS is that the predictor variables having the greatest effect on the models were identified as the MVN and SBC instructions. A performance comparison of the design in this paper was made with similar designs where it was shown that the approach of the design in this work yielded more robust models with a reported error rate of 0.000629193 for logic power and 0.023096787 for data power.

INDEX TERMS Data power, IoT, logic power, power prediction model.

I. INTRODUCTION

The era of green technologies and the drive towards a net-zero world are two important realities that are playing central roles in the development of current and future technologies. A net-zero world powered by technologies is a universally accepted human aspiration strongly undergirded by different resolutions from several UN bodies [1], [2], [3] and Conference of the Parties (COP) 26 [4] as a response to the dual challenge of global warming and rapidly changing climate that pose an existential threat to the future of humanity. To realize this aspiration, several technologies like internet-of-things (IoT) have emerged with the aim of creating the possibility of efficient management of environmental resources and grid power consumption in real-time in order to curb wastages that negatively affect the environment. Green technologies like eco cars, smart buildings, smart agriculture etc., being an integral part of the solution themselves, rely heavily on IoT for their interconnectivity and optimal functionality [5]. Thus, IoT is indirectly the center of gravity for technological solutions geared towards a net-zero future.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Quan.

IoT is a microcontroller-driven technology that has a central processor on each of its node in a given network [6], [7], [8]. Girded with this information, the development of apps that run on IoT nodes must be in a manner that optimally utilizes the capabilities of the processors at the lowest possible power cost in order to assure long-term operations especially in remote locations. One way to achieve this is for the processor to be able to predict the rate of power utilization, and supply such an information to a running app that then uses it to decide the best possible route to take in reducing power consumption. For this undertaking to be meaningful, one must decide what type of processor to target for the design of a power prediction model; this is important because there are billions of IoT nodes having different types of processors (from a number of chip makers and vendors) in existence today [9], [10], [11]. A reasonable approach will be to target a processor that is widely used in IoT nodes; a candidate processor will be the CortexM series family of processors from Advanced RISC Machines (ARM). The choice of these ARM processors is hinged on the fact that they have the most significant share of the market for IoT-powered technologies as shown in figure 1 [12], [13].

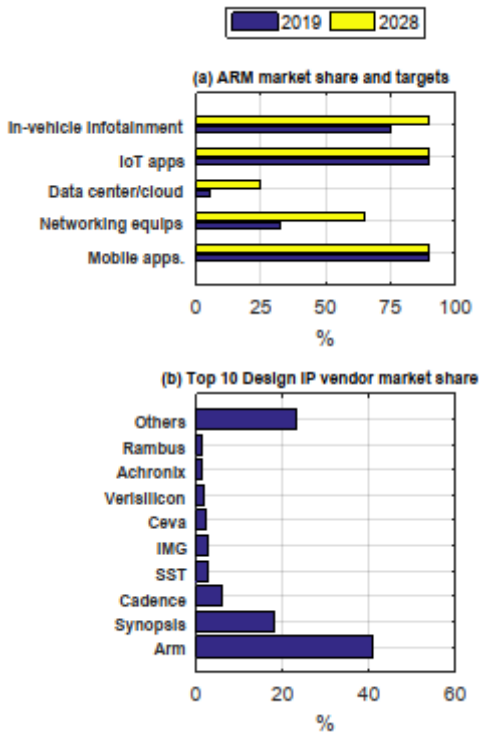


FIGURE 1. Market share of ARM processors.

Having established a reasonable basis for the choice of a target processor, the second important factor is the choice of technique to use for estimating the processor power. There are quite a number of techniques for the estimation of processor power utilization [8], [14], [15], [16]; of these, the instruction level technique aligns naturally with the requirements for the design a power prediction model because it allows the complete isolation of the estimated power of an application from the hardware [8]. This is a desirable feature for any focused-driven power estimation technique. Based on this understanding, the design in this work will be an instruction level power estimation technique.

To set the tone for the design, the paper is organized as follows- section II presents an overview of related works that have been done by other authors on instruction level power estimation. Section III presents the methodology for the design of the prediction model. In section IV, various results obtained by simulations based on the methodology are shown, alongside a performance comparison to similar designs. A conclusion and open issues about future research directions are finally discussed in section V.

II. OVERVIEW OF RELATED WORKS

Quite a number research on instruction level power estimation have been conducted by different authors. A few of these include the work by [17] where an instruction level energy model for VLIW architectures was used for high level power exploration framework. Specifically, they focused on the reduction of the complexity of the energy model of k-issue VLIW processor with respect to the number of operations from $O\left(|\text{ISA}|^k\right)$ to $\left(O\left(k * |\text{ISA}|^2\right)\right)$.

The instantaneous current drawn by the processor during the execution of instructions was used by [18] as a basis for the design of a technique for modeling instruction level energy of pipelined processors. Using an appropriate instrumentation, energy costs were modeled with regards to a reference instruction. The costs factored inter-cycle energy components that cancel out when summed to produce the total estimates of a given program.

An instruction level power estimator called IPEN was developed by [19] for sensor networks. To be robust, IPEN was designed independent of a host OS; it is therefore capable of estimating different types of sensor node software. The operation of IPEN is based on the profiling of peripheral accesses and function calls when an application is running.

By the combination of an empirical method and a statistical analysis method technique, [20] developed a method for accurately deriving the energy consumption model at instruction level. The equation developed by their technique works in such a manner that for any given program, the energy behavior is characterized by observing the properties of the instructions in the program.

Measurement based instruction level power analysis was used as a basis by [21] for the development of a technique for costing the power utilization of software in an accurate and practical way. The technique was applied to three commercial processors that are architecturally different. The systematic analysis by their technique lead to the identification of sources of software power consumption that can be targeted during software design and transformation techniques.

Apart from these attempts, several other authors recently used instruction level technique for the estimation of

processor power utilization. A summary of some of these is shown in Table 1.

TABLE 1. Recent research efforts in instruction level power estimation.

Author(s)/ Year	Title of work	Methodology	Strength	Weakness
Kiran & Ravish / 2021 Ref: [22]	Instruction Based Power Estimation Method in AURIX Microcontroller	The technique is based on the execution of instruction on an automotive microcontroller called AURIX. The estimation used some set of instructions that AURIX uses for data transfers and memory storage.	The creation of trace data and current measurement from the trace buffer provided a significant degree of coherency in the quality of data used for the estimation.	There was no justification given for the choice of selected instructions used as a basis for the estimator.
Kulkarni & Udupi / 2019 Ref: [23]	Instruction Level Energy Consumption Estimation of Embedded Processor	A technique was developed for the estimation of performance and energy utilization by ARM Cortex M4 processor. Three different methods were used for the measurement of software energy consumption	The inclusion of the effects of inter-instruction and static power makes the technique robust and more accurate than similar techniques that do not factor the effects of these parameters	A sample rate of 200ksps was chosen for all measurements irrespective of their peculiarity; it is not clear how this value was arrived at. To find the base cost, each instruction was executed 1000 times; for large number of instruction types, this approach can be very costly.
Wang & Zwolinski / 2014 Ref: [24]	An Improved Instruction-Level Power Model for ARM11 Microprocessor	An instruction level power model based on ARM11 processor was designed for predicting power of software applications. The model does not treat every instruction individually, and does not factor the effects of two adjacent instructions.	The model uses significantly less number of input data to achieve good results.	Results that are less accurate than similar models are likely to be produced because inter-instruction effects is not factored by this technique.
Bazzaz, M.,et al / 2013 Ref: [25]	An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems	A model was designed, and whose parameters included instruction opcode, number of shift operations, register bank bit flips, Hamming distance, instruction weights, and memory access type	The model is a very robust model because it encompasses a significant number of factors not considered by similar designs.	The model ignored the system initialization part and assumed that the system always operates in its main part; for a system with large number of peripherals, the likely error due to this assumption can be significant

III. METHODOLOGY

The approach in this work is divided into five stages and spanning a number of tools from Open Virtual Platforms (OVP), to MATLAB, and to VIVADO. Figure 2 shows the flow for the methodology using these tools.

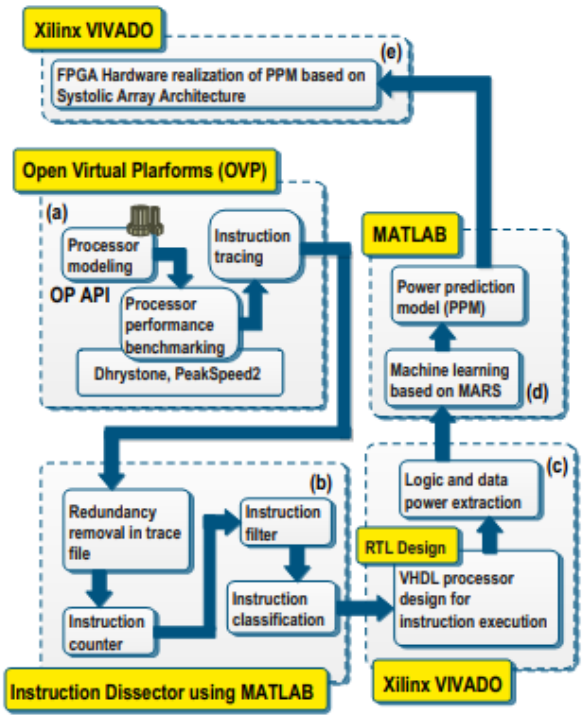


FIGURE 2. Workflow for methodology.

The first stage in the methodology is the modeling of the Cortex M processor using OVP. This is shown in figure 2a,

and it involves three sub-stages i.e. processor modeling, performance benchmarking, and instruction tracing. The processor modeling involves the use of OP API to design and integrate three key components of the processor considered in this work i.e. the bus, the CPU, and the memory. The interconnectivity of these components form the measurement setup for the work. Figure 3 shows the schematic representation of the interconnectivity; the components are integrated using OP API.

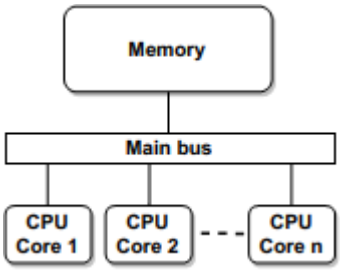


FIGURE 3. Measurement setup based on OVP model of processor integrated components.

The performance of the integrated system is tested using selected benchmarks like Dhrystone, and PeekSpeed2 as shown in figure 2a. The instructions used in executing the benchmarks by the processor are then extracted using instruction tracing by the OP API. A code algorithm for the operations in figure 2a is shown as follows:

Algorithm 1 Code Algorithm for Processor Modeling, Benchmarking, and Instruction Tracing

Input: *CPUCores*, *MainBus*, *RAM*, *Benchmark*
Output: *InstructionTrace*
1: Connect processor core to *MainBus*
2: Connect processor to *RAM*
3: Load *Benchmark n*
4: **while** *Benchmark n* **do**
5: initialize instruction pointers
6: run *Benchmark* on processor
7: $CPUId \leftarrow \{x \mid x \in CPUCores\}$
8: $PointerAddress \leftarrow \{x \mid x \in InstructionsAddresses\}$
9: $InstructionHexCode \leftarrow \{x \mid x \in InstructionsHexCodes\}$
10: $InstructionName \leftarrow \{x \mid x \in InstructionsNames\}$
13: **end while**
14: $InstructionTrace \leftarrow (CPUId, PointerAddress, InstructionHexCode, InstructionName)$

Figure 4 shows a partial view of the inputs and anticipated output from Algorithm 1; a partial view of the OP APIs used from processor modeling to instruction tracing are shown.

The second stage as shown in figure 2b is the dissector of the instructions that were run by the processor while executing benchmarks in figure 2a. The first step is the removal of redundancies in the instruction trace files; this process extracts only the listed instructions in the trace file. The extracted instructions are then counted in the instruction counter step to determine how many times each instruction was executed during the benchmark operations. The information from the instruction is fed to the instruction filter step; this step will separate instructions that were not executed thus having a count value of zero, from the instructions that were executed and thus having a count value greater than zero. The executed instructions are fed to the instruction classification step where they are classified into instruction categories as specified by ARM i.e. the designers of the processors; the reader is referred to [26] for further information on instruction groups for instruction classification.

Algorithm 2 shows the pseudocode for the operation of the instruction dissector using MATLAB; figure 5 presents in more conceptual details the operations that happen in the instruction dissector stage.

In figure 2c i.e. the third stage of the methodology, a design of a RISC processor based on ARM instruction set architecture is performed; the design is a general purpose processor whose decoder is tailored to execute the instructions that were actually executed during the benchmarking operations; for the work considered in this paper, the decoder is designed to execute data processing instructions. This approach ensures that redundancy is eliminated by having a decoder that does not have hardware for instructions that will not considered; hence, an accurate power utilization information will become obtainable by this approach.

Algorithm 2 Code Algorithm for Instruction Dissection

Input: InstructionTrace
Output: InstructionClassification
//Redundancy removal
1: InstrName ← {x | x ∈ InstructionTrace and x is all instruction names in InstructionTrace}
2: Name ← InstrName
3: j ← 0
// Instruction counter
4: **while** Name **do**
5: Count ← 0
6: **for all**k = 0 **to** N − 1 **do**
7: **if**(Name_j == InstrName_k) **then**
8: **if** (($\text{Name}_j \cap$ {x | x ∈ Name and $\text{Name}_0 \le x \le \text{Name}_{j-1}$}) == 0) **then**
9: Count ← Count + 1
10: **else**
11: Count ← Count
12: **end if**
13: **end if**
14: **end for**
15: NameCount_j ← Count
16: j ← j + 1
17: **end while**
// Instruction filter
18: ExeInstruction ← {x | x ∈ Name and NameCount of Name > 0 }
19: NonExeInstruction ← {x | x ∈ Name and NameCount of Name == 0 }
// Instruction classification
20: InstrType ← {x | x ∈ ExeInstruction and ExeInstruction is sorted into instruction type}
21: InstrTypeCount ← $\sum_{x \in S} x$ where S ← {y | y is the count value for each instruction in a selected InstrType}
22: InstrClassification ← (InstrType, InstrTypeCount)

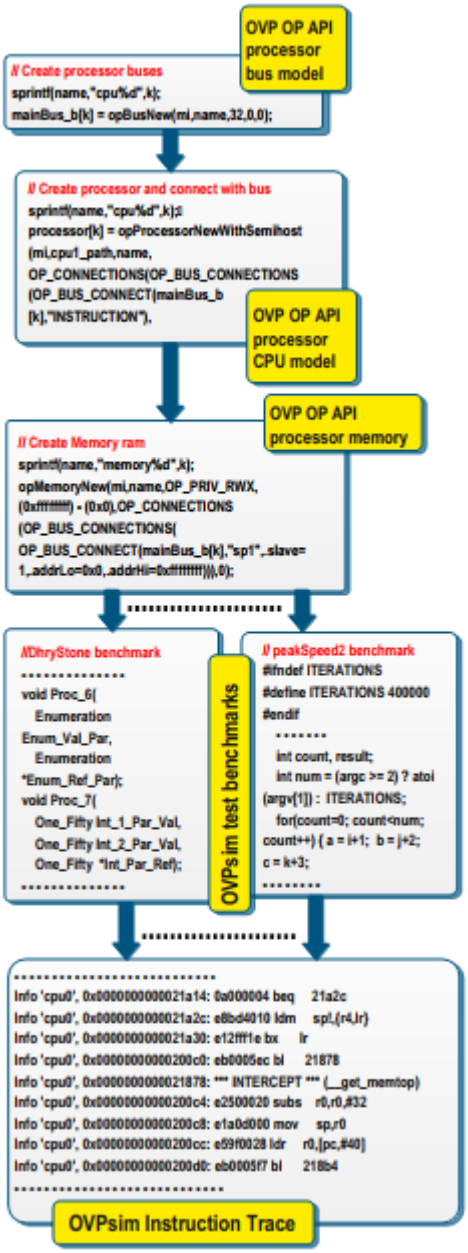


FIGURE 4. Processor modeling, benchmarking, and instruction tracing using OP APIs.

VHDL was used as the coding language for the hardware description of the processor, and Xilinx VIVADO was used as the coding environment. The instruction decoder in the processor design in this stage is a modification of the decoder in the processor design by [27]; all other components used for the processor were adopted from the VHDL description of the components by [27]. The operation of the processor is tested by executing each data processing instruction that was actually executed by the processor model designed using OVP in figure 2a; for each execution, the logic and data power utilization of the instruction type is extracted using the Switching Activity Interchange Format (SAIF) file. The SAIF file contains the static and dynamic power utilization of an instruction class; this work focuses on the dynamic power utilization because it is what an app developer can readily optimize. Figure 6 shows the conceptual details of the workflow activities in figure 2c.

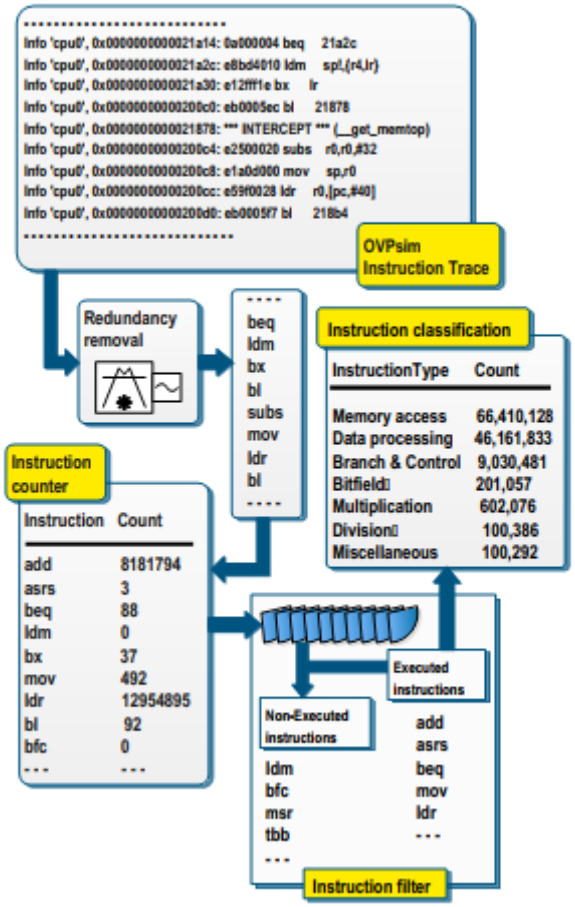


FIGURE 5. Operation of dissector for instruction dissection.

The fourth stage in the methodology i.e. figure 2d is machine learning stage; it churns the logic and data power utilization by computing the sum of power utilization for each row of data. This yields an input/output relationship dataset that is fed to the multivariate adaptive regression splines (MARS) as a machine learning technique.

MARS is a data-driven machine learning technique that uses a non-parametric regression approach to break a training dataset into different splines- piecewise linear segments characterized by differing slopes. The segments are delimited using knots in such a manner that the sub-divisions between any adjacent regions are marked to obtain basis functions (BFs) which are piecewise curves [28], [29], [30], [31]. Mathematically, MARS can be expressed as [30]:

$$y = C_0 + \sum_{i=1}^{N} C_i \prod_{j=1}^{k_j} b_{ji}\left(x_{\vartheta(j,i)}\right) \quad (1)$$

where y is the output variable, C_0 a constant, C_i the vector of coefficients associated with non-constant BFs, $b_{ji}\left(x_{\vartheta(j,i)}\right)$ is the BF with truncated power and having $\vartheta\ (j,\ i)$ as the index of independent variable used in the i^{th} term of the j^{th} product, and k_j being a parameter that limits the interaction order. Any spline b_{ji}, can be defined by the following relationships [28], [29], [30], [31]:

$$b_{\mathrm{ji}}\ (x) = \left|x - t_{\mathrm{ji}}\right|_+^q = \begin{cases} \left(x - t_{\mathrm{ji}}\right)^q \ if\ \ x < t_{\mathrm{ji}} \\ 0\ if\ \ x \geq t \end{cases}$$
$$b_{\mathrm{ji+1}}\ (x) = \left|t_{\mathrm{ji}} - x\right|_+^q = \begin{cases} \left(t_{\mathrm{ji}} - x\right)^q \ if\ \ x < t_{\mathrm{ji}} \\ 0\ if\ \ x \geq t \end{cases} \quad (2)$$

where t_{ji} is the knot of the spline, $q\ (q > 0)$ represents the spline power and the degree of smoothness of the resultant function approximation. The determination of the basis function to be included in the model is done by generalized cross-validation (GCV); this is the mean of the squared residual error divided by a penalty whose complexity depends on the model and is mathematically expressed as [30]:

$$\mathrm{GCV} = \frac{\frac{1}{N}\sum_{i=1}^{N}\left[y_i - f\ (x_i)\right]^2}{\left[1 - \frac{M + d \times (M-1)/2}{N}\right]^2} \quad (3)$$

where M is the number of bases functions, d the penalty for each basis function in the sub-model, N the number of data sets, and $f\ (x_i)$ the predicted values by MARS.

The output from the machine learning using MARS is a power prediction model (PPM) of the form:

$$y = C_0 + C_1\mathrm{BF}_1 - C_2\mathrm{BF}_2 \pm \cdots \pm C_n\mathrm{BF}_n \quad (4)$$

Algorithm 3 shows the pseudocode for the machine learning based on MARS, and Algorithm 4 shows the pseudocode for the power prediction model.

Algorithm 3 Pseudocode for Machine Learning Based on MARS

Input: *InstrPower* from RTL design
// InstrPower is all instruction power in RTL design
Output: *PMEquation*
1: *DataPower* ← {$x \mid x \epsilon$ all data power in *InstrPower*}
2: *LogicPower* ← {$x \mid x \epsilon$ all logic power in *InstrPower*}
3: *TrainingDataPower* ← {$x \mid x \epsilon$ *DataPower* and 80% of *DataPower*}
4: *TestingDataPower* ← {$x \mid x \epsilon$ *DataPower* and 20% of *DataPower*}
5: *TrainingLogicPower* ← {$x \mid x \epsilon$ *LogicPower* and 80% of *LogicPower*}
6: *TestingLogicPower* ← {$x \mid x \epsilon$ *LogicPower* and 20% of *LogicPower*}
7: *DataPowerModel* ← aresbuild (*TrainingDataPower*)
8: *LogicPowerModel* ← aresbuild (*TrainingLogicPower*)
// aresbuild is the function that executes MARS
9: *DataPowerModelValidation* ← *DataPowerModel* (*TestingDataPower*)
10: *LogicPowerModelValidation* ← *LogicPowerModel*(*TestingLogicPower*)
11: *DPModelEquation* ← areseq (*DataPowerModel*)
12: *LPPowerModelEquation* ← areseq (*LogicPowerModel*)
// aresbuild is the function that produces the model equation used for PPM
13: *PMEquation* ← (*DPModelEquation*, *LPModelEquation*)

Figure 7 shows the conceptual workflow in the fourth stage i.e. the operations in figure 2d.

The last stage of the methodology is shown in figure 2e; it involves an FPGA-based design of a systolic array

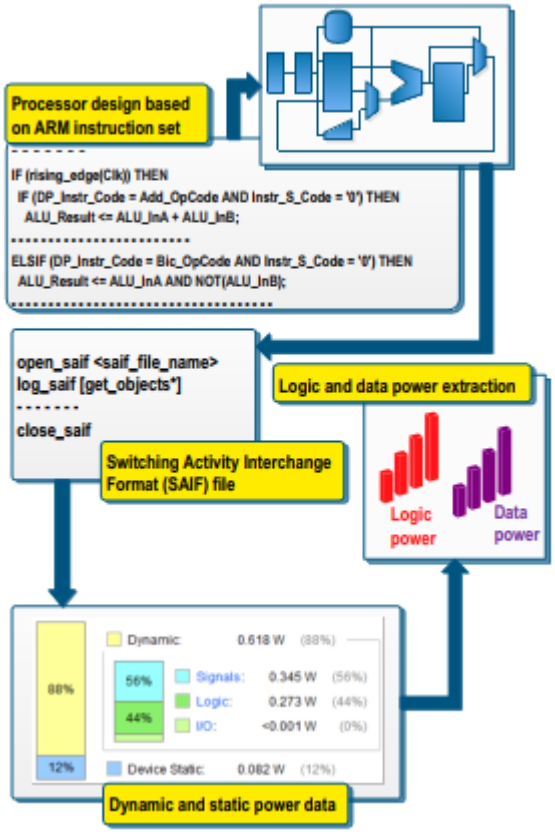


FIGURE 6. ARM based processor design and power utilization extraction.

Algorithm 4 Pseudocode for Power Prediction Model

Input: *PowerModelEquation*
Output: *PowerPredictionModel*
1: *DataPowerEquation* ← *DataPowerModelEquation* ϵ *PMEquation*
2: *LogicPowerEquation* ← *LogicPowerModelEquation* ϵ *PMEquation*
3: $bft1 \leftarrow \begin{cases} x - t, x > t \\ 0, otherwise \end{cases}$
4: $bft2 \leftarrow \begin{cases} t - x, x < t \\ 0, otherwise \end{cases}$
// *bft* implies Basis Function Type
5: *bfdp* ← *DataPowerEquation*(*bft1*, *bft2*)
// bfdp is basis functions for data power
6: *bflp* ← *LogicPowerEquation*(*bft1*, *bft2*)
// bflp is basis functions for logic power
7: *DataPowerPredictionModule* ← $\{x \mid x \epsilon\ bfdp$ and $bfdp_1 \leq x \leq bfdp_N\}$
8: *LogicPowerPredictionModule* ← $\{x \mid x \epsilon\ bflp$ and $bflp_1 \leq x \leq bflp_M\}$
9: *PowerPredictionModel* ← (*DataPowerPredictionModule*, *LogicPowerPredictionModule*)

architecture based on the PPM equation. The systolic array architecture will used as a sub-circuit that can be embedded in an overall design will be used for predicting power utilization for an application running on Cortex-M processor.

IV. ANALYSIS OF RESULTS

This section presents an analysis of the results obtained from the designs performed in this works, as well as from the methodology through which the designs were made. The

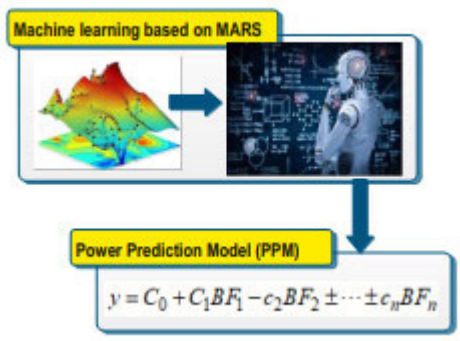


FIGURE 7. Machine learning and power prediction model.

analysis is constructed based on the results obtained from the generalized workflow in figure 2.

A. INSTRUCTION TRACING ANALYSIS

The instruction tracing is the first result obtained from the design flow process; it is the final output from the modeling of the processor and the execution of the selected test benches. A partial view of the traced instructions is shown in Table 2 for 15 samples from the trace file after executing the PeakSpeed2 benchmark.

TABLE 2. Partial view of executed instructions in the trace file.

S/N	Information	Address	Opcode	Instruction
1	Info 'cpu0'	0x0000000000020098:	e3a0b000	mov fp,#0
2	Info 'cpu0'	0x000000000002009c:	e3b07000	movs r7,#0
3	Info 'cpu0'	0x00000000000200a0:	e10f0000	mrs r0,CPSR
4	Info 'cpu0'	0x00000000000200a4:	e3c000c0	bic r0,r0,#192
5	Info 'cpu0'	0x00000000000200a8:	e129f000	msr CPSR_fc,r0
6	Info 'cpu0'	0x00000000000200ac:	e3b01000	movs r1,#0
7	Info 'cpu0'	0x00000000000200b0:	e59f0038	ldr r0,[pc,#56]
8	Info 'cpu0'	0x0000000000020164:	e28db004	add fp,sp,#4
9	Info 'cpu0'	0x00000000000200b8:	e0522000	subs r2,r2,r0
10	Info 'cpu0'	0x00000000000200bc:	eb000621	bl 21948
11	Info 'cpu0'	0x0000000000021948:	e3100003	tst r0,#3
12	Info 'cpu0'	0x000000000002194c:	e92d4010	stmdb sp!,{r4,lr}
13	Info 'cpu0'	0x0000000000021984:	e3520003	cmp r2,#3
14	Info 'cpu0'	0x0000000000024088:	0284cf53	addeq ip,r4,#332
15	Info 'cpu0'	0x000000000002016c:	e50b0010	str r0,[fp,#-16]

The information column in Table 2 indicates that only a single CPU called "cpu0" was modeled; the address column shows the respective memory addresses from where the executed instructions were fetched. The opcode column show the machine code (in hex) for executing an instruction, while the instruction column indicate the actual instruction that the CPU executed. Table 3 shows the types of benchmarks executed and the total number of instructions traced for each benchmark.

B. INSTRUCTION CLASSIFICATION ANALYSIS

Instruction classification as shown in figure 2b is the output of the instruction dissector. Its task is to determine the total number of instructions executed for every category of instructions. The categories were selected based on the

TABLE 3. Benchmarks and numbers of traced instructions.

Benchmark	Total instructions traced
PeakSpeed2	135,002,730
DhryStone	122,606,253
Fibonacci	138,147,711

categories in [26] as earlier indicated. Table 4 shows the output of the instruction dissector; it can be seen that the PeekSpeed2 has the highest number of memory access instructions while the Fibonacci test bench has the highest number of data processing instructions.

TABLE 4. Instruction categorization and count by dissector.

Instruction Category	Number of Executed Instructions		
	PeakSpeed2	DhryStone	Fibonacci
Data Processing	45,001,412	46,161,833	67,534,306
Memory Access	90,000,813	66,410,128	67,427,482
Multiply	98	602,076	2,086
Divide	28	100,386	596
Bitfield	47	201,057	938
Branch and Control	330	9,030,481	3,182,301
Miscellaneous	2	100,292	2
Total instructions	135,002,730	122,606,253	138,147,711

It should be noted that the derived instruction categories from the instruction dissector is the basis on which the instruction decoder for the VHDL design of the processor in figure 2c is performed. For the work in this paper, data processing instructions are the focus for the design of the prediction for power utilization. The prediction model of other instruction classes will be considered in a future work.

C. LOGIC AND DATA POWER EXTRACTION

The extraction of the logic and data power for each instruction type is performed after the SAIF process; for each instruction type, the logic and data power are extracted using the power analysis tool in the VIVADO design suite. Figure 8 shows an example of extracted power the SUB instruction, and the EOR instruction. In figure 8, the power associated with signals represents the data power. The input/output (I/O) power is not considered in this work as it is negligible with a value of less than 0.001W. As indicated in figure 8, for some instructions the data power is higher than the logic power, while for some others, the reverse is the case.

The data and logic power for all the executed instructions are extracted and used as the basis for the construction of a table that will be used by the machine learning stage. For each instruction category, the instruction dissector, as shown in figure 5, filters out the executed instructions from the non-executed instructions. Consequently, for the data processing instructions, the executed instructions, as determined by the instruction dissector, are shown in Table 5 in the context of their description as predictor variables, and their definition. Thus, eleven predictor variables were derived from the

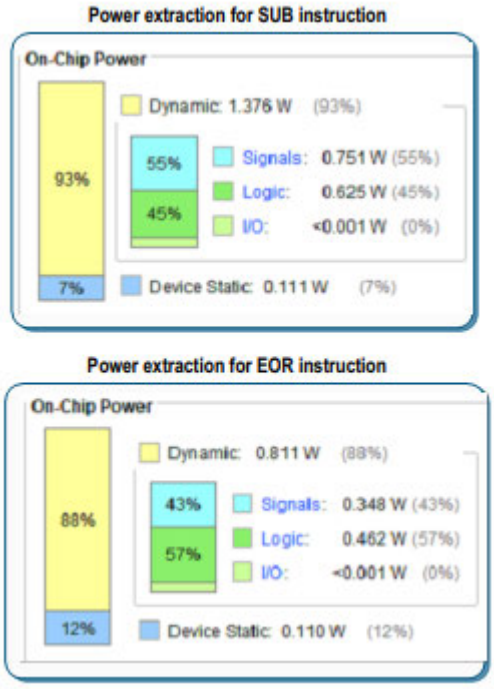


FIGURE 8. Extracted power for SUB and EOR instructions.

dissection of the data processing instructions executed by the processor. The same variable nomenclature holds for data power, and logic power; hence, their definition applies to all two categories of power type. For the data processing instructions, the sample data for the logic and data power are shown in Table 6.

TABLE 5. Summary of input and output variables for data power.

	Description	*Definition*
Predictor variables (Inputs)	x1	ADD instruction data/logic power
	x2	SUB instruction data/logic power
	x3	AND instruction data/logic power
	x4	BIC instruction data/logic power
	x5	CMP instruction data/logic power
	x6	MOV instruction data/logic power
	x7	RSB instruction data/logic power
	x8	EOR instruction data/logic power
	x9	SBC instruction data/logic power
	x10	ORR instruction data/logic power
	x11	MVN instruction data/logic power
Output variables	Data Power, Logic Power	

D. MACHINE LEARNING ANALYSIS

Using MARS, the optimum number of basis for the logic power, and data power for data processing instructions in Tables 6 are selected as shown in figures 9 and 10 respectively. The optimum number of basis was estimated by Generalized Cross Validation (GCV) and cross validation converged on the same value of 7 as the optimum number of basis for the logic, and data power; this is a desirable property and a good indicator of the quality of the data from the perspective of stability.

A key feature of MARS is in its ability to identify what predictor variables are having the strongest effect on a derived model. Consequently, for the logic and data power, the analysis of the weights of their predictor variables is presented in Tables 7 and 8 respectively.

In Table 7, there are three predictor variables which have an effect on the logic power model, with x11 representing the MVN instruction having the strongest effect; following it is x5 representing the CMP instruction having a lesser effect. The predictor variable x8 representing the EOR instruction

TABLE 6. Sample training and testing data.

Logic power											Data power										
Sample testing data											Sample training data										
x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
0.00372	0.00608	0.00412	0.00443	0.00504	0.00372	0.00547	0.00664	0.00528	0.00466	0.00373	0.00372	0.00606	0.00302	0.00388	0.00502	0.00279	0.00546	0.00634	0.00522	0.00344	0.00279
0.00359	0.00607	0.00374	0.00393	0.00504	0.00359	0.00547	0.00634	0.00523	0.00396	0.00359	0.00332	0.00574	0.00292	0.00388	0.00486	0.00260	0.00529	0.00634	0.00498	0.00341	0.00260
0.00345	0.00606	0.00359	0.00388	0.00503	0.00332	0.00547	0.00634	0.00522	0.00395	0.00335	0.00279	0.00574	0.00228	0.00388	0.00486	0.00248	0.00529	0.00634	0.00498	0.00271	0.00205
0.00210	0.00573	0.00180	0.00386	0.00486	0.00247	0.00529	0.00579	0.00493	0.00209	0.00185	0.00260	0.00573	0.00180	0.00386	0.00486	0.00247	0.00529	0.00579	0.00493	0.00209	0.00185
x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
0.01018	0.02979	0.00210	0.02019	0.02672	0.01215	0.02839	0.03306	0.02598	0.00210	0.00744	0.00969	0.02836	0.00075	0.01924	0.02536	0.01158	0.02700	0.03150	0.02471	0.00075	0.00709
0.00970	0.02843	0.00075	0.01969	0.02546	0.01202	0.02705	0.03306	0.02502	0.00075	0.00731	0.00867	0.02762	0.00004	0.01924	0.02354	0.01157	0.02497	0.03150	0.02436	0.00026	0.00684
0.00970	0.02836	0.00075	0.01924	0.02542	0.01158	0.02704	0.03150	0.02472	0.00075	0.00730	0.00710	0.01615	0.00000	0.00822	0.01713	0.00388	0.01546	0.00700	0.01689	0.00026	0.00290
0.00210	0.01578	0.00000	0.00822	0.01712	0.00384	0.01545	0.00700	0.01519	0.00026	0.00266	0.00710	0.00578	0.00000	0.00822	0.01712	0.00384	0.01545	0.00700	0.01519	0.00026	0.00266

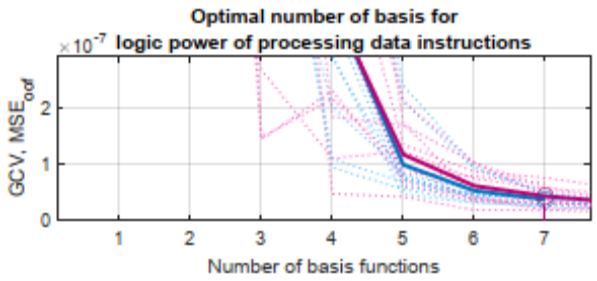


FIGURE 9. Optimum number of basis for logic power.

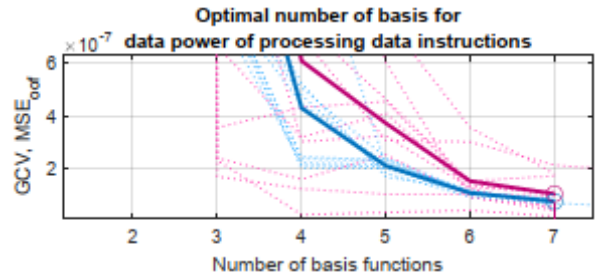


FIGURE 10. Optimum number of basis for data power.

TABLE 7. Weights of logic power predictor variables.

Variable	delGCV	nSubsets	subsRSS	subsGCV	
1	0.0000	0	0.000	0.000	unused
2	0.0000	0	0.000	0.000	unused
3	0.0000	0	0.000	0.000	unused
4	0.0000	0	0.000	0.000	unused
5	95.949	5	100.0	100.0	
6	0.0000	0	0.000	0.000	unused
7	0.0000	0	0.000	0.000	unused
8	38.995	3	0.079	0.079	
9	0.0000	0	0.000	0.000	unused
10	0.0000	0	0.000	0.000	unused
11	100.00	1	0.015	0.015	

TABLE 8. Weights of data power predictor variables.

Variable	delGC	nSubset	subsRS	subsGC	
1	0.0000	0	0.000	0.000	unused
2	95.056	1	0.007	0.006	
3	0.0000	0	0.000	0.000	unused
4	0.0000	0	0.000	0.000	unused
5	0.0000	0	0.000	0.000	unused
6	0.0000	0	0.000	0.000	unused
7	81.209	4	100.0	100.0	
8	0.0000	0	0.000	0.000	unused
9	100.00	3	2.071	2.096	
10	0.0000	0	0.000	0.000	unused
11	0.0000	0	0.000	0.000	unused

has the least effect on the model for predictor variables that are used; other predictor variables are unused. Table 9 shows the basis function for this model having the participating predictor variables.

TABLE 9. Basis functions with corresponding logic power model equations.

BF	Equation	BF	Equation
BF1	max (0, x5 - 0.001078)	BF4	max (0, 2.181e-07 - x8)
BF2	max (0,0.001078 - x5)	BF5	max (0, x11 – 5.782e-06)
BF3	max (0, x8 – 2.181e-07)	BF6	max (0, 5.782e-06 – x11)
Output Equation			
*y = 0.004551 +5.93*BF1 -3.733*BF2 +1.928*BF3 -1761*BF4 +2.107*BF5 -25.21*BF6*			

In Table 8, the variable x9 that represents the SBC has the strongest effect on the data power model; following it is *x2* that represents the SUB instruction and then the RSB instruction represented by x7; other predictor variables are unused. Table 10 shows the basis function and the corresponding equation for the data power model.

TABLE 10. Basis functions with corresponding equations for data power model.

BF	Equation	BF	Equation
BF1	max (0, x7 – 5.002e-08)	BF4	max (0, x2 – 0.0002021)
BF2	max (0, x9 – 0.0005452)	BF5	max (0, 0.0002021 - x2)
BF3	max (0, 0.0005452 - x9)		
Output Equation			
*y = 0.0007622 +2.041*BF1 +0.8434*BF2 -1.038*BF3 +1.292*BF4 -0.9726*BF5*			

The plot for the logic power model shown in figure 11was plotted with x5 and x11 as independent variables, and y as the dependent variable. The non-linearity effect of the independent variables on the output of the model is visible along the x5 axis. In a similar analysis, the plot of the data model shown in figure 12 has x2 and x9 as the independent variables, and y as the dependent variable. Figure 12 shows better linear relationship between the independent variables and the dependent variable. The variation in linearity between figure 11 and 12 will affect the rate of convergence of the BFs associated with the models.

E. BASIS FUNCTION CONVERGENCE ANALYSIS

The performance of the basis functions for the data power and logic power associated with the data processing instructions is shown in figure 13a and 13b respectively. It can be observed that the BFs of the data power achieved convergence earlier than those of the logic power. This is expected because for any given set of data processing operations, the data involved are likely to remain constant except for a few intermediate values, while the logic operations on the data can span from a few operations to a large number of operations. As an example, consider the circuit shown in figure 14; it can be observed that while there are three input data associated with its operations, there are however seven logic operations that occur between that data. These kind of scenarios have an effect on

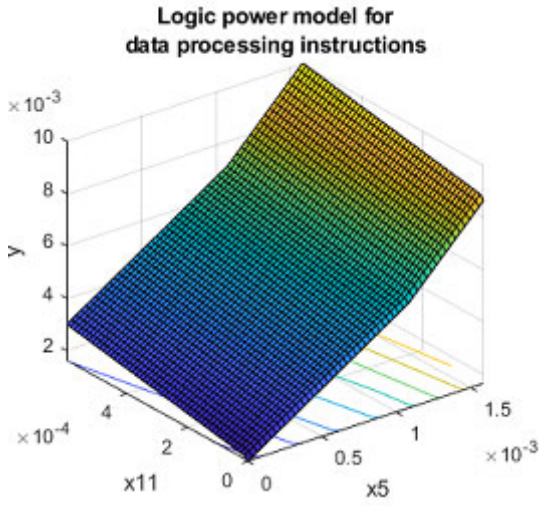


FIGURE 11. Linearity plot of logic power model.

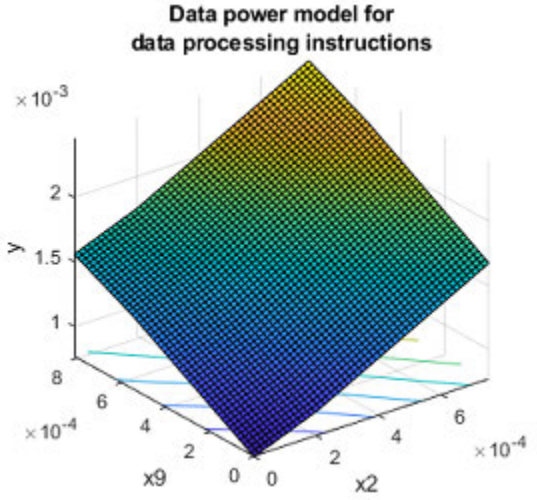


FIGURE 12. Linearity plot of data power model.

the latency associated with the convergence of the logic power BFs.

F. COEFFICIENT OF DETERMINATION (R^2) ANALYSIS

To determine the proportion of variance of the dependent variables (power outputs from the model equations) that can be predicted from the independent predictor variables, the R^2 results for the two models is presented in this regard in Table 11.

TABLE 11. R^2 values for power model categories.

Parameter	Logic power	Data power
R^2	0.9995	0.9996

The results in Table 11 indicate that the MARS models for the power model categories of data processing instructions indicate that only marginal differences will be observed

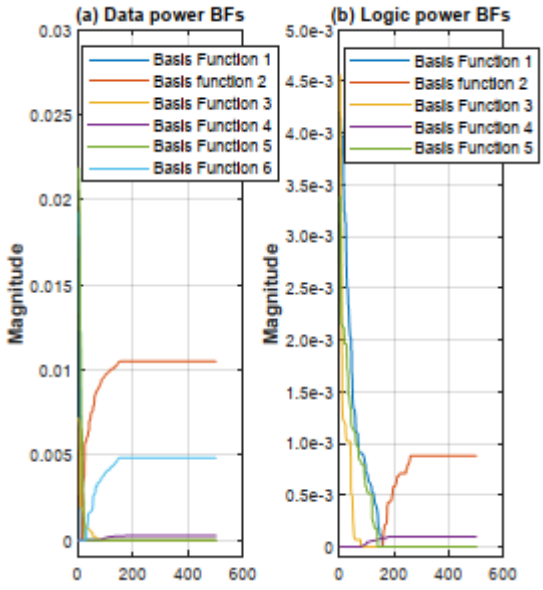


FIGURE 13. Performance of BFs.

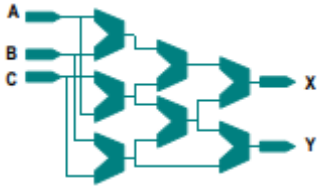


FIGURE 14. Data points and logic operations in a circuit.

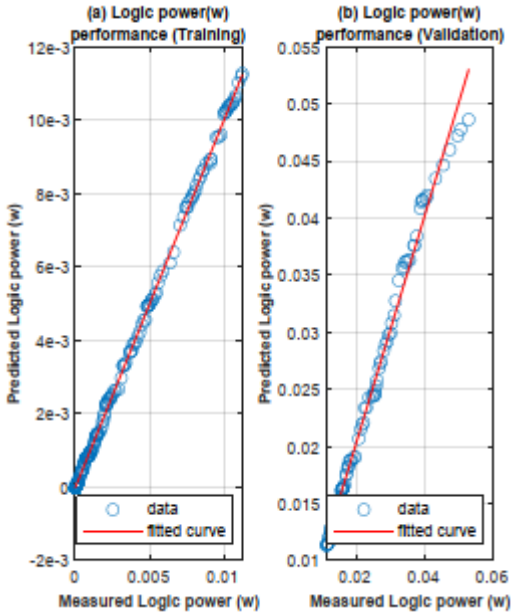


FIGURE 15. Curve fitting performance for logic power.

between measured power and the predicted power. For the logic power and data power, the prediction model will be able to predict power utilization with an error of 0.05%

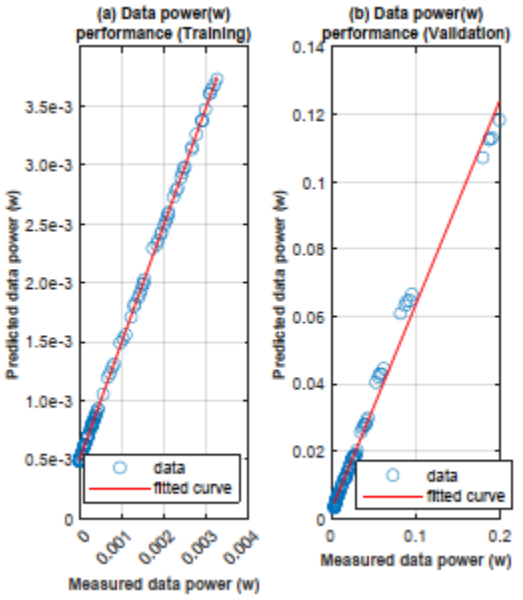


FIGURE 16. Curve fitting performance for data power.

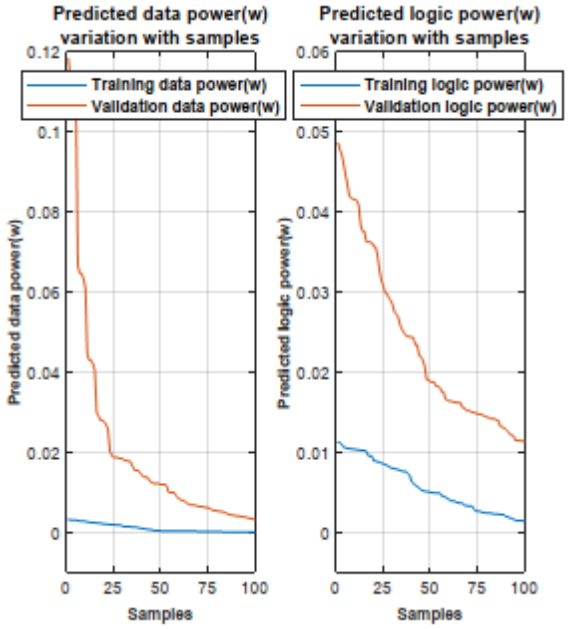


FIGURE 17. Convergence performance between training and validation phase.

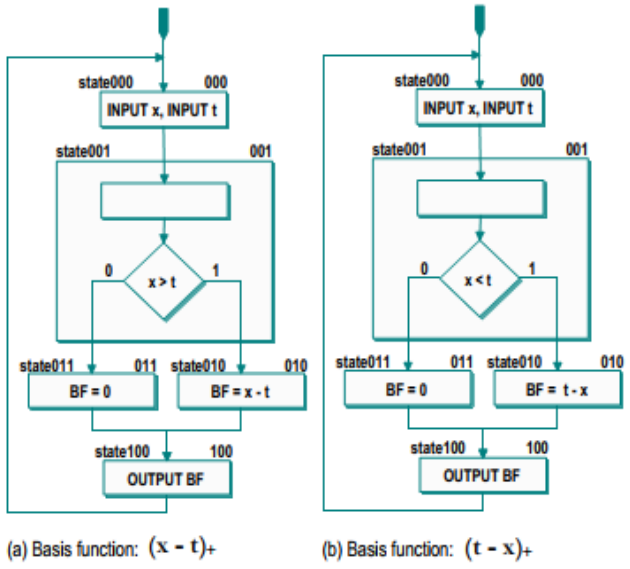


FIGURE 18. Design of basis functions using ASM chart.

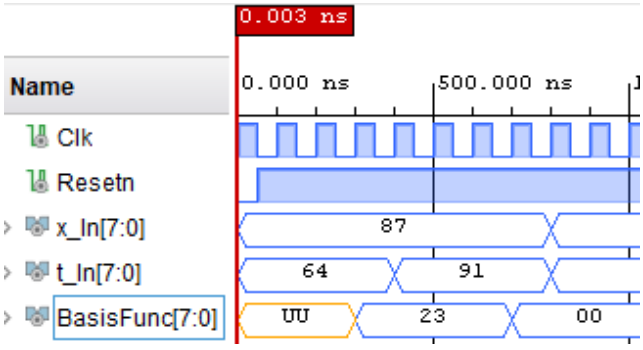


FIGURE 19. $(x - t)_+$ simulation. Note: all values are in HEX notation.

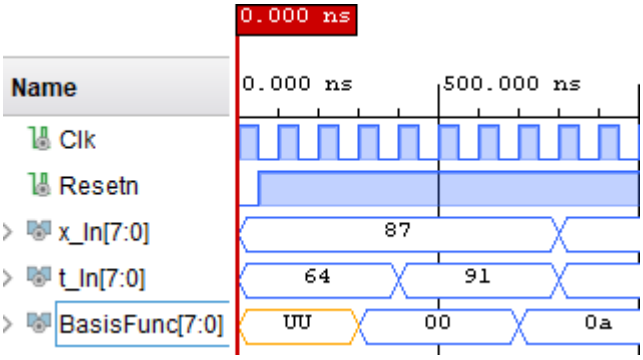


FIGURE 20. $(t - x)_+$ simulation. Note: all values are in HEX notation.

and 0.04% for logic power and data power respectively. This level of performance implies that the prediction models are reliable; hence, there is a justification for the design of an embedded sub-circuit based on these models. Such a circuit when integrated in a system running on the Cortex M3 processor can be saddled with the responsibility of continuously predicting the logic power and data power for data processing operations. With such an information, an application running on the system would be able to make optimal decisions during data operations to guarantee energy efficiency while at the same time maintaining a good level of performance.

To corroborate the level of accuracy, figure 15 shows the plot of the performance of the logic power for the training phase and validating phase. For the training phase, an almost perfect fit was achieved by the model in figure 15a, while during the validation phase, a slight deviation from the

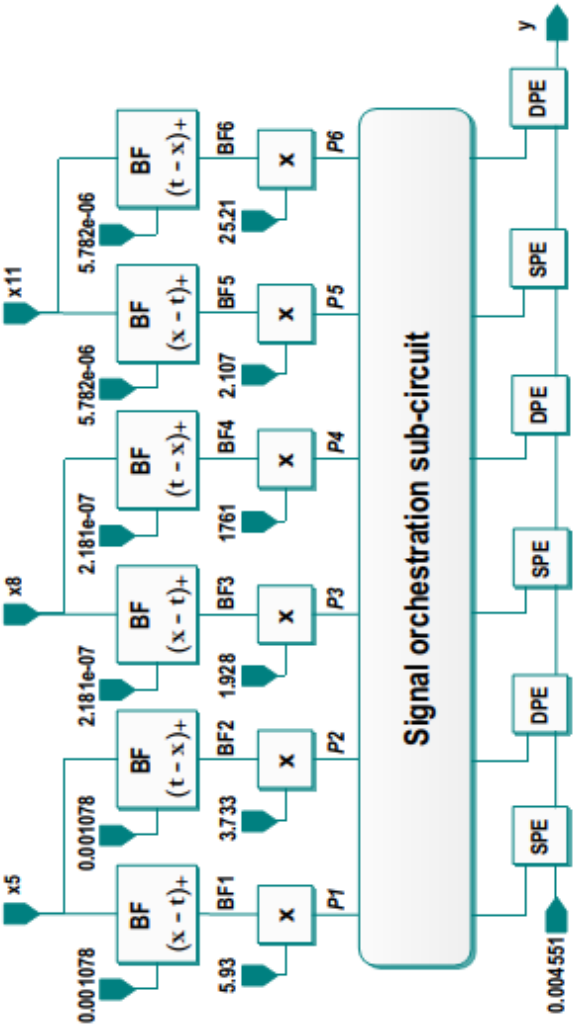


FIGURE 21. Systolic array circuit for FPGA realization of logic power model.

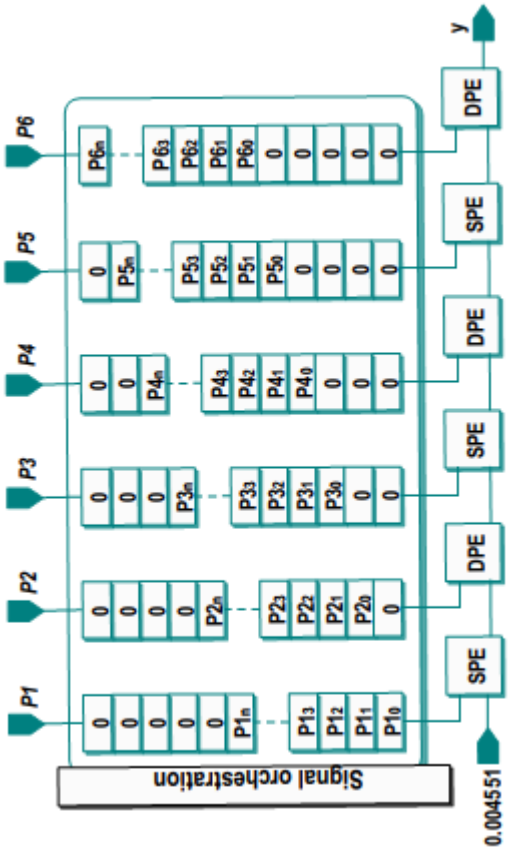


FIGURE 22. Logic power orchestration sub-circuit.

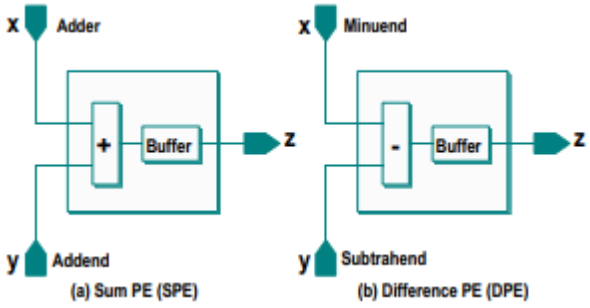


FIGURE 23. Circuits for SPE and DPEl.

fitting curve occurred at about 0.04W. In a similar analysis, figure 16 shows the curve fitting performance for data power. Figure 16a shows a better fitting in the training phase for the data power than the training phase for the logic power in figure 15a. Similarly, for the validation phase, the data power model showed better curve fitting in figure 16b than the logic power model in figure 54b. A convergence plot between the predicted power during the training phase, and the predicted power during the validation for the case of the data power and logic power models is shown in figure 17, and as expected a faster convergence is observed in the case of the data power model than that of the logic power model. The reason for the convergence rates is a function of the curve fitting performance in figure 15 and 16.

G. FPGA REALIZATION FOR POWER PREDICTION MODEL (PPM)

The equations for the logic power model in Table 9 and the data power model in Table 10 can be used for the hardware realization of the PPMs. A systolic array approach can be used for such a design since the independent variables can be viewed as concurrent streams of data. For the sake of brevity to avoid over-blotting of this work, the reader is referred to [32], [33], [34], and [35] for details on systolic arrays and design techniques. The choice of an FPGA for such a realization naturally comes to play because FPGAs have concurrent hardware that makes them ideal for systolic circuit design.

A natural take off point for the FPGA realization is the design of the BFs, and based on the expressions in (2), the BFs are either of the form $(x - t)_+$ or $(t - x)_+$. Figure 18 shows the design of the BFs using an Algorithmic State Machine (ASM)

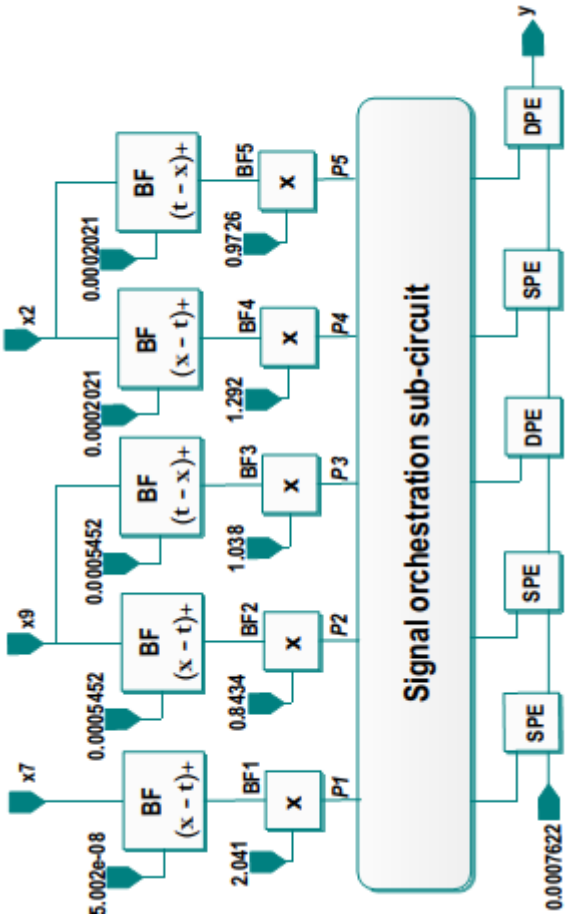


FIGURE 24. Systolic array circuit for FPGA realization of data power model.

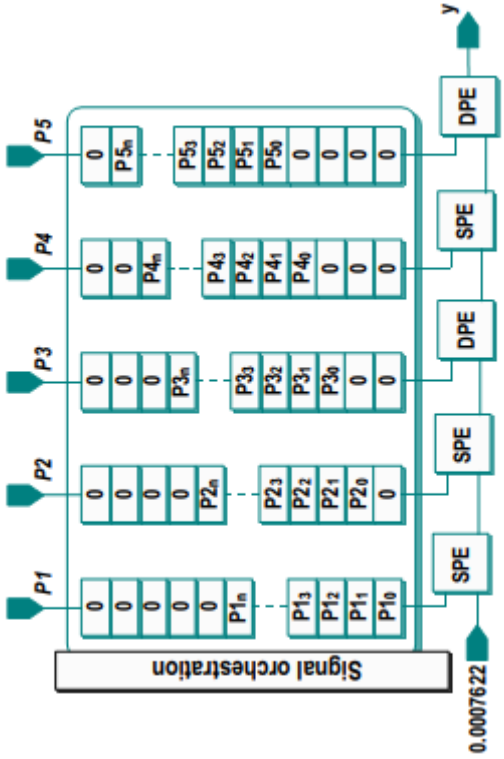


FIGURE 25. Data power orchestration sub-circuit.

Chart. VHDL was used for the hardware description design of the BFs; the design was a generic design capable of handling data with different bit width, and figure 19 and 20 show the simulation for the BFs for a case of an 8-bit bit width data.

Using the output equation in Table 9, a systolic array based circuit for the FPGA realization of the logic power model is shown in figure 21.

The signal orchestration sub-circuit in the design ensures proper timing of the signals arriving to the SPEs (Summation Processing Element) and DPEs (Difference Processing Element) so that accurate results are obtained by ensuring all the requirements for data dependencies are satisfied. Figure 22 shows the signal orchestration sub-circuit, and figure 23 shows the circuits for the SPE and DPE blocks.

Similarly, figure 24 shows the FPGA realization for the data power model and figure 25 shows its corresponding signal orchestration sub-circuit.

The simulation for the design of FPGA realization of the logic, and data power models is shown in figure 26 for the first four of the results. It should be noted that the results are fixed point results because the FPGA realizations were designed on the basis of fixed point representation.

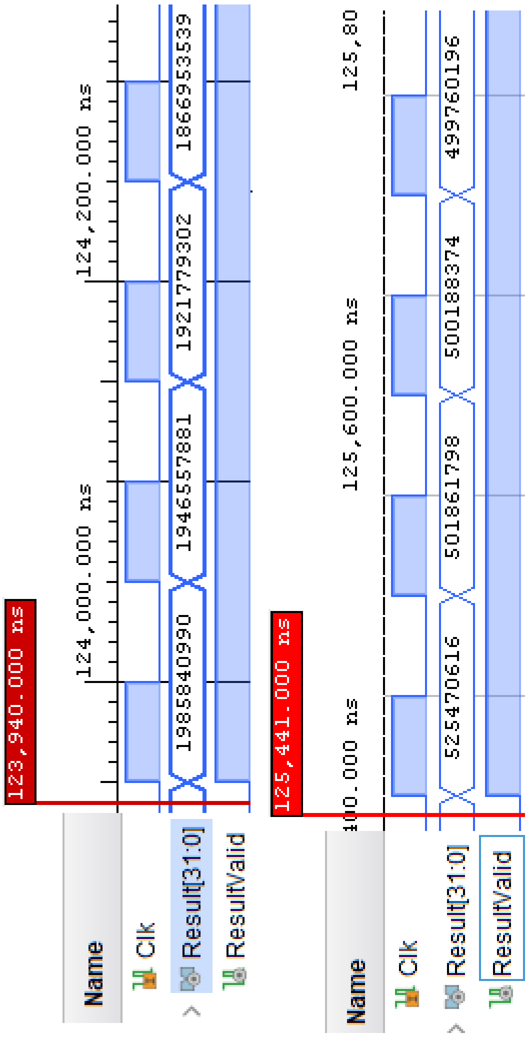
To verify the functional accuracy of the FPGA realizations, two models were developed for the architecture in figure 21 using MATLAB and Ada programming; similarly, two models were developed for the architecture in figure 24 using MATLAB and Ada programming. Table 12 shows the first seven samples of the logic and data power results respectively from the FPGA realization using VHDL, the MATLAB model, and the Ada model.

A Root Mean Squared Error (RMSE) analysis was performed between the results from the FPGA realization via VHDL and the corresponding models in MATLAB and Ada using the relationship in (5) as defined by [36] as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left[y\,(i) - \hat{y}\,(i)\right]^2}{N}} \tag{5}$$

where $y\,(i)$ is the expected output (obtained from the MATLAB and Ada models), $\hat{y}\,(i)$ is the actual output (obtained from the FPGA realization via VHDL), and N is the number of samples.

From the logic power results, the RMSE between the MATLAB model and FPGA realization is 0.000000, and the RMSE between the Ada model and FPGA realization is 6.29193e-6, which translates to 0.000629193% error. Similarly, from the data power results, the RMSE between the MATLAB model and FPGA realization is 0.000000, and the RMSE between the Ada model and the FPGA realization



(a) Logic power (b) Data power

FIGURE 26. First four samples of predicted power by processor. Note: Computed values are in fixed-point representation.

is 2.30968e-4, which translates to 0.023096787% error. It can be observed that the FPGA realization results show very high accuracy from the RMSE values; the RMSE values between the Ada model and FPGA realizations is due to the strong typed nature of Ada as a language; it thus is able to represent data with a degree of accuracy that outperforms modeling languages like MATLAB. For this reason, Ada is the language of choice in safety-critical systems from avionics to nuclear plant control.

V. PERFORMANCE COMPARISON WITH RELATED WORKS

A performance comparison between the design in this paper and similar designs by other authors is shown in Table 13. The comparison focused on robustness of design, and accuracy. It can be observed the design in this paper compares favorably with other designs. The design methodology in this paper had features like predictor variable identification and mini hardware realization; these features make the design in this paper more robust in comparison to other designs.

The predictor variable identification feature made the design in this work capable of identifying the particular instruction that consuming the most power; such an information is important is very important in optimal design of IoT applications that are anticipated to operate in remote locations for an extended period of time. As reported in Table 13, a very high degree of accuracy was achieved by the prediction mini hardware realization of the circuits of the prediction

TABLE 12. First seven samples of logic and data power results.

Logic power results						Data power results					
VHDL		MATLAB		Ada		VHDL		MATLAB		Ada	
Fixed Point	Floating Point	Fixed Point	Floating Point	Fixed Point	floating Point	Fixed Point	Floating Point	Fixed Point	Floating Point	Fixed Point	floating Point
1985840990	29.5913367	1985840990	29.59133670	1985840990	29.5913367	525470616	7.830122352	525470616	7.830122352	525470616	7.830122352
1946557881	29.00597276	1946557881	29.00597276	1946557881	29.00597276	501861798	7.478323549	501861798	7.478323549	501861798	7.478323549
1921779302	28.63674316	1921779302	28.63674316	1921779302	28.63674316	500188374	7.453387588	500188374	7.453387588	500188374	7.453387588
1866953539	27.81977563	1866953539	27.81977563	1866953539	27.81977563	499760196	7.447007239	499760196	7.447007239	499760196	7.447007239
1806059327	26.91238116	1806059327	26.91238116	1806059327	26.91238116	475695302	7.088412374	475695302	7.088412374	475695302	7.088412374
1753678004	26.13183862	1753678004	26.13183862	1753678004	26.13183862	294584432	4.389650106	294584432	4.389650106	294584432	4.389650106
1687302098	25.14276055	1687302098	25.14276055	1687302098	25.14276055	285974600	4.261353612	285974600	4.261353612	285974600	4.261353612

TABLE 13. Performance comparison.

S/N	1	2	3	4	5	6	7
Ref.	[22]	[23]	[37]	[24]	[25]	[38]	Proposed work
Year	2021	2019	2017	2014	2013	2008	2024
Title	Instruction Based Power Estimation Method in AURIX Microcontroller	Instruction Level Energy Consumption Estimation of Embedded Processor	A simulation framework for code-level energy estimation of embedded soft-core processors	An Improved Instruction-Level Power Model for ARM11 Microprocessor	An Accurate Instruction-Level Energy Estimation Model and Tool for Embedded Systems	High-Level Estimation of Execution Time and Energy Consumption for Fast Homogeneous MPSoCs Prototyping	Development of a model for prediction of IoT Processor Power Utilization using Instruction Dissection Based Technique
Mini hardware realization from model	No	No	No	No	No	No	Yes
Instruction tracing	Yes	Yes	No	No	No	No	Yes
Instruction grouping	No	Yes	Yes	Yes	Yes	Yes	Yes
Predictor variable identification	No	No	No	No	No	No	Yes
Branch / inter instruction consideration	No	Yes	Yes	No	Yes	No	No
Average power / energy estimation error (%) (Absolute value)	1.12	3.788611	2.0	4.88	3.68	7.05	0.000629193 for logic power 0.023096787 for data power

TABLE 14. Performance comparison with earlier works.

S/N	Year	Technique	Priori knowledge of input statistic	Resolution	Redundant information processing	Instruction grouping	Exact instruction analysis	Number of design assumptions	Granularity
1	2019	Learning-Based CPU Power Modeling	No	Not Reported	Yes	No	No	1	Mixed - RTL level & Source code level
2	2017	A Statistical Approach to Power Estimation for x86 Processors	No	Good – Equation based	Yes	Yes	No	1	Function level
3	2014	Expert system based modeling	No	Moderate – combines LUTs and mathematical	Yes	Yes	No	6	Function level
4	2009	Power macro modeling	Yes	Low – relies on LUT only	Yes	Yes	No	1	Gate level
5	2005	Power Estimation Strategies For A Low-Power Security Processor	No	Not Reported	Yes	Yes	No	4	Gate level
6	2002	Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach	No	Not Reported	Yes	Yes	No	4	System level
7	1998	Software power estimation and optimization	No	Good – Bquation based	Yes	Yes	No	2	Source code level
8	2024	Proposed work	No	Good – Equation based	No	Yes	Yes	1	Mixed - RTL level & Instruction level

TABLE 14. *(Continued.)* **Performance comparison with earlier works.**

Portability	No	Yes	No	No	No	No	No	Yes
Scalability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reference	[39]	[40]	[41]	[42]	[43]	[44]	[45]	

model in this work. The accuracy was arrived at using RMSE between the prediction model that was coded in MATLAB and Ada, and the FPGA realization based on systolic array architecture via VHDL. The work by authors as reported in Table 13 also had good accuracy; it should be noted the basis for the determination of their accuracy is not the same as the approach in this work.

A comparative analysis is also presented in Table 14 for performance comparison between the technique presented in this work, and the techniques published in earlier work based on a number of parameters that can be broadly applied.

VI. CONCLUSION

Power prediction models for processors are poised to become an integral part of future designs of applications that are envisaged to be energy-aware. Consequently, the design of processor power models using different techniques has become an active area of research. The design in this paper presented an instruction level approach for the design of a CORTEX M power prediction using MARS. The data for the design were extracted from the power analysis tool of VIVADO where all the power associated with the operation of the processor during the execution of an instruction can be obtained. The design was able to identify the predictor variables like MVN and SBC instructions having very strong effect on the derived models. The accuracy of the models was underscored by the curve fitting performance where strong convergence was observed. The design also proposed a mini hardware realization of the prediction model based on a systolic array architecture that was coded and functionally verified using VHDL. The performance comparison of the design in this work was compared to similar designs in the context of robustness and accuracy; the design in this work compared favorably.

CONFLICT OF INTEREST DECLARATION

The authors hereby declare the following for the manuscript titled ''Development of a Model for Prediction of IoT Processor Power Utilization using Instruction Dissection Based Technique'' that:

They have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

- ☒ The article they have submitted to the journal for review is original, has been written by the stated authors, and has not been published elsewhere.
- ☒ The images that they have submitted to the journal for review are original, was taken by the stated authors, and has not been published elsewhere.
- ☒ This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.
- ☒ They have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript

S/N	Author's Name	Affiliation
1	Peter Yusuf Dibal	Computer Engineering Department, University of Maiduguri, Nigeria
2	Hashim Eltahir Ahmed	King Khalid University, Saudi Arabia
3	Elizabeth Nonye Onwuka	Telecommunication Engineering Department, Federal University of Technology Minna, Nigeria
4	Suleiman Zubair	Telecommunication Engineering Department, Federal University of Technology Minna, Nigeria

REFERENCES

[1] (2023). *UN Group Adopts 2050 Goal of Net-zero Emissions From Planes*. Accessed: Aug. 10, 2023. [Online]. Available: https://apnews.com/article/travel-business-air-pollution-c61aeed41e6abc825caadebaa8e9d469

[2] (2023). *Net-Zero Emissions Must Be Met By 2050 or COVID-19 Impact on Global Economies Will Pale Beside Climate Crisis, Secretary-General Tells Finance Summit*. Accessed: Aug. 10, 2023. [Online]. Available: https://press.un.org/en/2020/sgsm20411.doc.htm

[3] UNDP. (2023). *NET ZERO PATHWAYS*. Accessed: Aug. 10, 2023. [Online]. Available: https://climatepromise.undp.org/what-we-do/areas-of-work/net-zero-pathways

[4] (2021). *COP26: Together for Our Planet*. Accessed: Aug. 10, 2023. [Online]. Available: https://www.un.org/en/climatechange/cop26

[5] P. Dibal, E. Onwuka, Z. Suleiman, B. Salihu, E. Nwankwo, and S. Okoh, "An overview of IoT solutions in climate smart agriculture for food security in sub Saharan Africa: Challenges and prospects," *EAI Endorsed Trans. Internet Things*, vol. 8, no. 3, p. e1, Sep. 2022.

[6] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Microprocessor optimizations for the Internet of Things: A survey," 2016, *arXiv:1603.02393*.

[7] M. A. El-Razek, M. B. Abdelhalim, and H. H. Issa, "Dynamic power reduction of microprocessors for IoT applications," *J. Adv. Veh. Syst.*, vol. 2, no. 1, pp. 10–20, 2016.

[8] P. Y. Dibal, E. N. Onwuka, S. Zubair, E. I. Nwankwo, S. A. Okoh, B. A. Salihu, and H. B. Mustaphab, "Processor power and energy consumption estimation techniques in IoT applications: A review," *Internet Things*, vol. 21, Apr. 2023, Art. no. 100655.

[9] (2019). *Number of Internet of Things (IoT) Connected Devices Worldwide From 2019 To 2023, With Forecasts From 2022 To 2030*. Accessed: Aug. 10, 2023. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/
[10] (2023). *IoT Connections Outlook*. Accessed: Aug. 10, 2023. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook
[11] S. Sinha. (2023). *State of IoT 2023: Number of Connected IoT Devices Growing 16% To 16.7 Billion Globally*. Accessed: Aug. 10, 2023. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/
[12] (2019). *Arm's Market Share and Targets Across Key Technology Markets in 2019 and 2028 Fiscal Years*. Accessed: Aug. 10, 2023. [Online]. Available: https://www.statista.com/statistics/1132112/arm-market-share-targets/
[13] (2019). *Top 10 Design IP Vendor Market Share Ranking*. Accessed: Aug. 12, 2023. [Online]. Available: https://www.yolegroup.com/product/report/design-ip/
[14] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019.
[15] V. A. Kulkarni and G. R. Udupi, "Instruction level power consumption estimation-issues and review," *J. Multidiscip. Eng. Sci. Technol.*, vol. 4, no. 2, pp. 6776–6781, 2017.
[16] H. Sultan, G. Ananthanarayanan, and S. R. Sarangi, "Processor power estimation techniques: A survey," *Int. J. High Perform. Syst. Archit.*, vol. 5, no. 2, p. 93, 2014.
[17] A. Bona, M. Sami, D. Sciuto, C. Silvano, V. Zaccaria, and R. Zafalon, "Reducing the complexity of instruction-level power models for VLIW processors," *Design Autom. Embedded Syst.*, vol. 10, no. 1, pp. 49–67, Mar. 2005.
[18] S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis, and S. Blionas, "Instruction level energy modeling for pipelined processors," *J. Embed. Comput.*, vol. 1, pp. 317–324, May 2005.
[19] H. Joe, J. Park, C. Lim, D. Woo, and H. Kim, "Instruction-level power estimator for sensor networks," *ETRI J.*, vol. 30, no. 1, pp. 47–58, Feb. 2008.
[20] S. Lee, A. Ermedahl, S. L. Min, and N. Chang, "An accurate instruction-level energy consumption model for embedded RISC processors," *ACM SIGPLAN Notices*, vol. 36, no. 8, pp. 1–10, Aug. 2001, doi: 10.1145/384196.384201.
[21] V. Tiwari, S. Malik, A. Wolfe, and M. T.-C. Lee, "Instruction level power analysis and optimization of software," in *Proc. 9th Int. Conf. VLSI Design*, Jan. 1996, pp. 326–328, doi: 10.1109/ICVD.1996.489624.
[22] K. G. Shetty P S and D. R. Aradhya H V, "Instruction based power estimation method in AURIX micro controller," *J. Univ. Shanghai Sci. Technol.*, vol. 23, no. 6, pp. 784–793, Jun. 2021.
[23] V. A. Kulkarni and G. R. Udupi, "Instruction level energy consumption estimation of embedded processor," *Eur. J. Eng. Res. Sci.*, vol. 4, no. 2, pp. 40–44, 2019.
[24] W. Wang and M. Zwolinski, "An improved instruction-level power model for ARM11 microprocessor," in *High Performance Energy Efficient Embedded Systems*, 2014, pp. 1–7. [Online]. Available: https://eprints.soton.ac.uk/361481/1/hip3es_submission_3.pdf
[25] M. Bazzaz, M. Salehi, and A. Ejlali, "An accurate instruction-level energy estimation model and tool for embedded systems," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 7, pp. 1927–1934, Jul. 2013, doi: 10.1109/TIM.2013.2248288.
[26] (2010). *The Cortex-M3 Instruction Set*. [Online]. Available: https://users.ece.utexas.edu/~valvano/EE345M/CortexM3InstructionSet.pdf
[27] S. L. Harris and D. M. Harris, *Digital Design and Computer Architecture: ARM Edition*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
[28] M. Rezaie-Balf, "Multivariate adaptive regression splines model for prediction of local scour depth downstream of an apron under 2D horizontal jets," *Iranian J. Sci. Technol., Trans. Civil Eng.*, vol. 43, no. S1, pp. 103–115, Jul. 2019.
[29] P. Yuvaraj, A. R. Murthy, N. R. Iyer, P. Samui, and S. K. Sekar, "Multivariate adaptive regression splines model to predict fracture characteristics of high strength and ultra high strength concrete beams," *CMC*, vol. 36, no. 1, pp. 73–97, 2013.
[30] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, no. 1, pp. 1–67, Mar. 1991.
[31] P. Samui and D. P. Kothari, "A multivariate adaptive regression spline approach for prediction of maximum shear modulus and minimum damping ratio," *Eng. J.*, vol. 16, no. 5, pp. 69–78, Oct. 2012.
[32] J. R. Reinders, "Systolic Arrays," in *Encyclopedia of Parallel Computing*. Boston, MA, USA: Springer, 2011.
[33] M. Mukherjee and S. K. Sanyal, "2-D systolic array architecture of CBNS based discrete Hilbert transform processor," *Microprocessors Microsystems*, vol. 87, Nov. 2021, Art. no. 103509.
[34] A. Bigdeli, M. Biglari-Abhari, Z. Salcic, and Y. T. Lay, "A new pipelined systolic array-based architecture for matrix inversion in FPGAs with Kalman filter case study," *Eurasip J. Appl. Signal Process.*, vol. 2006, pp. 1–12, Apr. 2006.
[35] P. H. Langade and S. B. Patil, "Design of improved systolic array multiplier and its implementation on FPGA," *Int. J. Eng. Res. Gen. Sci.*, vol. 3, no. 6, pp. 838–845, 2015.
[36] T. O. Hodson, "Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not," *Geosci. Model Dev.*, vol. 3, pp. 1–10, May 2022.
[37] M. A. Pasha, U. Gul, M. Mushahar, and S. Masud, "A simulation framework for code-level energy estimation of embedded soft-core processors," *Simulation*, vol. 93, no. 10, pp. 809–823, Oct. 2017, doi: 10.1177/0037549717699074.
[38] S. J. Filho, A. Aguiar, C. A. M. Marcon, and F. P. Hessel, "High-level estimation of execution time and energy consumption for fast homogeneous MPSoCs prototyping," in *Proc. 19th IEEE/IFIP Int. Symp. Rapid Syst. Prototyping*, Jun. 2008, pp. 27–33, doi: 10.1109/RSP.2008.25.
[39] A. Krishna Ananda Kumar and A. Gerstlauer, "Learning-based CPU power modeling," in *Proc. ACM/IEEE 1st Workshop Mach. Learn. CAD (MLCAD)*, Sep. 2019, pp. 1–6, doi: 10.1109/MLCAD48534.2019.9142100.
[40] M. Chadha, T. Ilsche, M. Bielert, and W. E. Nagel, "A statistical approach to power estimation for ×86 processors," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2017, pp. 1012–1019, doi: 10.1109/IPDPSW.2017.98.
[41] B. P. Singh, A. Shankar, F. G. Wolff, and C. A. Papachristou, "An expert system based tool for pre-design chip power estimation," in *Proc. Des. Verification Conf.*, 2014, pp. 1–7.
[42] Y. A. Durrani and T. Riesgo, "Power estimation technique for DSP architectures," *Digit. Signal Process.*, vol. 19, no. 2, pp. 213–219, Mar. 2009.
[43] Y.-F. Lee, S.-Y. Huang, S.-Y. Hsu, I.-L. Chen, C.-T. Shieh, J.-C. Lin, and S.-C. Chang, "Power estimation strategies for a low-power security processor," in *Proc. ASP-DAC Asia South Pacific Design Autom. Conf.*, vol. 1, 2005, pp. 367–371.
[44] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, and M. Kandemir, "Using complete machine simulation for software power estimation: The SoftWatt approach," in *Proc. 8th Int. Symp. High Perform. Comput. Archit.*, Feb. 2002, pp. 141–150, doi: 10.1109/HPCA.2002.995705.
[45] J. T. Russell and M. F. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proc. Int. Conf. Comput. Design. VLSI Comput. Processors*, Oct. 1998, pp. 328–333, doi: 10.1109/ICCD.1998.727070.

DIBAL P. YUSUF received the master's degree in electronics and communications engineering from Teesside University, U.K., in 2011, and the Ph.D. degree in communication from the Federal University of Technology Minna, in 2019. His research interests include digital signal processing, wireless reconfigurable systems, quantum computing, and VLSI architectures for signal processing and communication systems.

HASHIM E. AHMED (Member, IEEE) received the Bachelor of Electronics Engineering degree in telecommunications engineering from Sudan University of Science and Technology (SUST), Sudan, in 2000, and the master's and Ph.D. degrees in electrical engineering (telecommunications engineering) from the Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), in 2009 and 2015, respectively. He was a Senior Researcher with the MIMOS Center of Excellence CoEx in Telecommunications Engineering, Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), and a member of the Telematic Research Group, from 2010 to 2015. He worked industrial experience at the MIMOS Berhad, National Applied Research and Development Centre, Kuala Lumpur, Malaysia, from December 2011 to December 2013. He has been an Assistant Professor with the Department of Computer Networks and Communications Engineering, King Khalid University (KKU), Saudi Arabia, since January 2016. His research interests include telecommunications engineering and networking engineering, satellite communications and radar systems, wireless sensor networks, the Internet of Things (IoT), 5G cellular systems, underground communication, software-defined networking, spectrum sharing using cognitive radio techniques (CR), geolocation database (GLD), LTE advanced, and TV white space (TVWS) management schemes.

ELIZABETH N. ONWUKA received the Bachelor of Engineering degree from the Electrical and Computer Engineering Department, Federal University of Technology (FUT) Minna, Niger State, Nigeria, and the Master of Engineering degree in telecommunications and the Ph.D. degree in communications and information systems engineering from Tsinghua University, Beijing, China. She is currently a Professor of telecommunications engineering. Her research interests include mobile communications networks, mobile IP networks, handoff management, paging, network integration, resource management in wireless networks, spectrum management, and big data analytics.

ZUBAIR SULEIMAN received the B.Eng. degree in electrical and computer engineering, in 2003, and the M.Eng. degree in communication engineering and the Ph.D. degree in wireless sensor networks from the Faculty of Electrical Engineering, MIMOS (Laboratory) Center of Excellence in Telecommunication Technology, Universiti Teknologi Malaysia, Johor.

He is a Lecturer with the Department of Telecommunication Engineering, Federal University of Technology Minna, Nigeria. His research interests include wireless sensor networks, embedded systems, data communication and networking, technological development in third-world countries, wireless communication, optical fiber communications, next-generation networks, and biomedical technology.