Universiteit Leiden

Master Computer Science

Early Design Stage Power Consumption Estimation of VLSI Circuits

Name:	Bingxuan Wu
Student ID:	s3284018
Date:	31/08/2023
Specialisation:	Advanced Computing & Systems
1st supervisor:	Dr. Todor Stefanov
2nd supervisor:	Dr. Nusa Zidaric

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Managing power consumption is crucial in the design of electronic devices. Power consumption analysis plays a significant role in optimizing performance, ensuring system reliability, and addressing security vulnerabilities. As integrated circuit designs continue to advance, early design stage power analysis and optimization have become essential for the development of agile and flexible very large-scale integrated circuit (VLSI). However, the existing early-power estimation approaches often face a trade-off between accuracy and efficiency.

We present a power model to estimate the power consumption of VLSI circuits at the `RTL` design stage. We employ a one-dimensional Convolutional Neural Networks (CNN) model for cycle-by-cycle power estimation and evaluate its performance on a lightweight cryptographic XOODYAK circuit. The power model is trained using a subset of the testbenches, capturing the relationships between `RTL` switching activity and corresponding gate-level power consumption. The model achieves a prediction accuracy exceeding 90% in 46.67% of the 106 different test cases, with an minimal NRMSE of 5.68%. Our approach is particularly well suited for application scenarios where power consumption needs to be estimated very fast under various workloads in the same design.

Contents

Chapter 1

Introduction

With the ever-increasing demand for high-performance and portable computing devices, power consumption has become a critical concern and a constraint factor in very large-scale integrated (VLSI) circuit design. For example, the peak power determines the thermal and electrical limits of a circuit design, which directly affects the performance, area, and cost of a computing system. Exceeding these constraints can lead to issues like overheating, efficiency reduction, or even system failures [1]. Additionally, although shrinking feature sizes present opportunities for increased complexity of integrated circuit (IC) designs, power consumption can significantly impact the ability to integrate more transistors on the chip as well as limit the feasible packaging and performance of VLSI circuits [2].

To better manage and optimize power consumption, early design stage power analysis has become an essential step in the modern IC design cycle. Accurate and efficient power estimation is required to meet the power specification requirements while mitigating risks and ensuring overall system performance. Gaining early insight into power consumption improves the flexibility and agility of the VLSI circuit development process, as potential power-related issues can be identified and optimized in a timely manner, thereby avoiding the costly redesign process.

Different approaches are employed to estimate power consumption at varying abstraction levels during the VLSI circuit design flow, as explained in Section 2.1. The design hierarchy, spanning from high-level specifications to low-level mask data, is visualized using the Y-Chart [3], as shown in Figure 1.1. It outlines the design refinement levels along three axes: functional, structural, and geometrical. The abstraction levels are well defined in [4], representing the design stages within the structural axis.

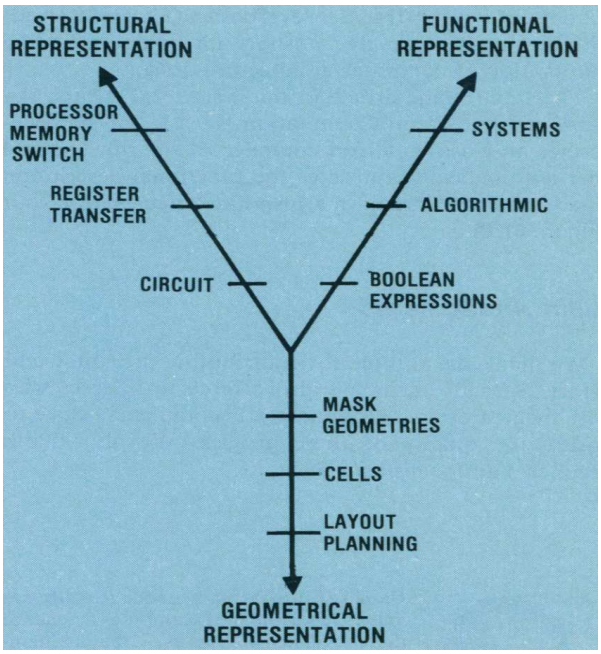


Figure 1.1: Tripartite Design Level Representations [3]

The lower abstraction level refers to the two bottom levels: gate level and transistor level. The conventional gate-level power estimation is supported by several commercial power solution tools, where gate-level simulation results and synthesized netlists are fed into these tools for power calculations [5]. The results obtained at the gate level or transistor level are typically more accurate due to the availability of detailed implementation technology information at these lower abstraction levels. However, gate-level simulation is known to be slow, typically processing only 10-1000 cycles per second [6]. Thus, it will hinder design efficiency and make them impractical for dynamic power optimization in large and complex circuit designs. Alternatively, some approaches use Register Transfer Level (`RTL`) simulation results and gate-level netlists, but cycle-based power consumption analysis using commercial power solution tools is still a computationally demanding and time-consuming task [7, 8].

At higher levels of abstraction in VLSI circuit design, namely architecture, algorithm, and system levels [9], power consumption estimation approaches can be broadly classified into two categories: analytical-based [10, 11] and regression-based methods [12, 13, 14, 15, 16]. However, these methods are less accurate because they lack low-level circuit details above the gate level. Moreover, these methods might be struggling

to capture or model the nonlinear relationships in the complex circuit systems [17].

The current power consumption estimation approaches face the trade-off problem between accuracy and efficiency. Given the dilemma between accurate but time-consuming gate-level simulations, and faster but less accurate analytical/regression-based methods at higher abstraction levels, a noteworthy research question arises: **How can we find the optimal 'sweet spot' for power consumption estimation by achieving a balance between estimation speed and high accuracy?**

In this thesis, we present a machine learning (ML) based power consumption model for power estimation of VLSI circuits at `RTL` design stage. Our goal is to leverage advanced ML techniques to efficiently estimate power consumption at a high abstraction level while retaining gate-level details of the VLSI circuits. To ensure cycle-accurate power consumption analysis, we used gate-level netlists and `RTL` switching activities as input for time-based power calculations to generate datasets. By capturing and learning the relationships between circuit switching activity and corresponding power consumption, the power estimation model can accurately infer the power consumption for any other input to the circuit.

Our modeling framework is particularly suitable for application scenarios where power consumption needs to be quickly estimated under different workloads on the same circuit design. Only `RTL` simulation is required as input of the trained model to provide accurate power consumption estimation, making it an efficient tool for assessing power consumption in VLSI circuits with various workload scenarios. The major contributions of this work are summarized as follows:

- Our approach provides an efficient power consumption estimation methodology by combining the advantages of the high-level abstraction of VLSI circuits with more detailed gate-level information. The framework enables power consumption analysis using switching activity information at the `RTL` level, making it a useful tool for early-stage power consumption estimation in the VLSI circuits design process. The power consumption model accelerates power estimation, which can be applied to estimate the power consumption for various user-specified workloads efficiently.

- We provide a cycle-by-cycle power consumption estimation model. The proposed cycle-accurate methodology facilitates detecting time intervals characterized by high or abnormal power consumption within the circuit. It enables designers to understand the power behavior of a circuit at the cycle level, which helps with power optimization and performance analysis.

- We explore the use of one-dimensional Convolutional Neural Networks (1D-CNN) model for time-based power consumption estimation. We evaluate the model on a lightweight cryptographic circuit (LWC) called XOODYAK, and demonstrate the capabilities of the proposed framework. For cycle-by-cycle power estimation with 106 different test cases for the XOODYAK circuit, 46.67% of the test cases show a model prediction accuracy exceeding 90%. The model achieves a minimal Normalized Root Mean Square Error (NRMSE) 5.1 of 5.68%.

The thesis is structured as follows: Chapter 2 and Chapter 3 introduce the background of this thesis and the related works. Chapter 4 presents the proposed methodology for power consumption estimation. Chapter 5 provides the experiments and results. We answer the research question and discuss the limitations of our approach as well as avenues for future work in Chapter 6. Chapter 7 gives an overall conclusion.

Chapter 2

Background

This chapter is intended to provide the background and prerequisite knowledge required to understand this thesis. We first introduce the design flow of the Application Specific Integration Circuits (ASIC), then emphasize the importance of early-stage power analysis in the circuit design process. Then we explain the sources of power consumption in the circuits. Subsequently, the top-level architecture of the Xoodyak circuit is presented, for which early power consumption estimation is carried out in our work. Finally, we briefly describe the 1D-CNN used for power consumption modeling in this thesis.

2.1 ASIC Design Flow

The process of designing an ASIC entails a series of steps. Each step is described in detail below. In order to achieve a successful ASIC design, designers need to explore the design space to determine optimal design parameters during the design flow, including performance, area, cost, and power consumption as well as ensure adherence to strict time-to-market constraints.

1) *Design entry & Functional Simulation:* RTL description is used for functional simulation to confirm the functional correctness of the circuit on the design entry level. It is expressed in hardware description languages such as Verilog and VHDL.

2) *Logic Synthesis:* The circuit netlist is generated using RTL and logic synthesis tools. The netlist is a description of the circuit in terms of library gates and their connections.

3) *Physical Design:* The gate-level netlist is further refined and transformed into a physical layout that takes into account the actual geometric placement and routing of components on the silicon chip. This phase includes floor planning, placement, and routing to ensure that the physical design meets performance, power, and area constraints.

4) *Postlayout Simulation & Signoff check:* This involves running simulations on the fully laid-out design to ensure its functionality matches the intended design specifications. After simulations, signoff checks such as Layout Versus Schematic (LVS) or Design Rule Checks (DRC) are performed to ensure the design adheres to manufacturing and design rules.

5) *Fabrication & Packaging & Testing:* Once the design has passed all simulations and verification steps, it's ready for fabrication. The designed ASIC is manufactured onto a silicon wafer through processes like lithography, etching, and doping. The resulting chips are then cut from the wafer, assembled into packages for protection and connectivity, and subjected to various tests to validate their functionality and performance.

The importance of performing early power analysis during the ASIC design flow lies in its ability to prevent costly redesign efforts. When power-related issues surface later in the design process during physical design or post-layout simulation, or even fabrication and testing, resolving these problems becomes increasingly difficult, expensive, and time-consuming [18]. To avoid such setbacks, it becomes crucial to estimate power consumption early in the logical synthesis phase. In addition, designers can make informed design decisions and timely adjustments to optimize power issues at a higher abstraction level, and explore the design space with more flexibility [11].

2.2 Power Consumption in the VLSI circuits

Power consumption in CMOS circuits arises from three main sources: 1) static leakage power, 2) short-circuit power, and 3) switching power. The leakage and short-circuit currents in CMOS circuits can be minimized by using appropriate device and circuit design techniques.

Switching power is the most significant contributor to the overall power consumption. It is the power dissipated when charging and discharging the capacitive load during state transitions [1], such as $V_{high} \rightarrow V_{low}$ (logic 1 $\rightarrow$ logic 0) or $V_{low} \rightarrow V_{high}$

(logic 0 → logic 1). The load includes the capacitance of the gate/cell input and the net capacitance of external downstream gates/cells. The average switching power is given by:

$$P_{\text{switching}} = \frac{1}{2} \cdot \alpha \cdot C_{\text{L}} \cdot V_{\text{dd}}^2 \cdot f_{clk} \tag{2.1}$$

where α is the switching activity factor, which is the average fraction of circuit nodes that change state per clock cycle. V_{dd} is the supply voltage, f_{clk} is the clock frequency, and C_L is the effective capacitance, representing the total capacitive load that is being charged and discharged during each state transition.

Therefore, when modeling the power consumption of CMOS circuits, the main focus is on the relationship between dynamic switching power consumption and state transitions, which are the key factors contributing to power consumption.

2.3 Lightweight Cryptographic Circuit

In this section, we first present the LWC hardware architecture designed of XOODYAK by [19] for the purpose of comparing cryptographic circuits submitted to the NIST LWC competition. We omit details on XOODYAK design, as they are not important to understand the work in this thesis, XOODYAK itself is implemented within the CryptoCore block, shown in Figure 2.1 [20].

The LWC core includes four essential units: `PreProcessor`, `CryptoCore`, `Header FIFO`, and `PostProcessor`. The interface of this LWC architecture consists of two input data buses, Public Data Inputs (`PDI`) and Secret Data Inputs (`SDI`), and an output data bus, Data Output (`DO`). `PDI` are inputs that are publicly accessible and typically include data such as the associated data and public message number. These inputs are provided to the `PreProcessor` units for further processing. `PDI` is divided into segments. Each segment begins with a segment header that describes its type and length. This approach ensures consistent communication between the sender and the LWC core, allowing for flexibility in managing different types of input data and ensuring that the LWC core correctly processes the provided data segments. `SDI` contains sensitive information that needs to be protected, such as the encryption or decryption key. `DO` is the output data bus through which the LWC core sends the processed data back to the calling entity. The processed data can include plaintext, ciphertext, and authentication tags.

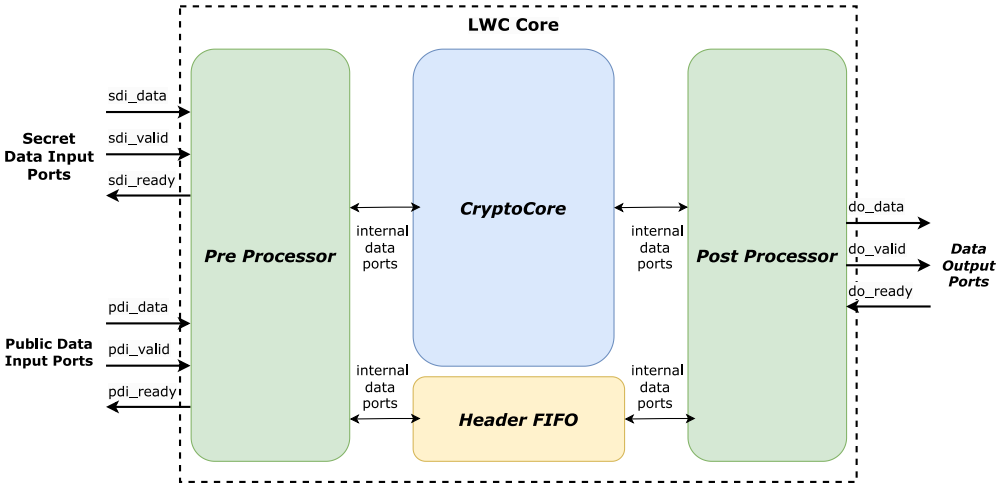


Figure 2.1: The interface and Top-level block diagram of the LWC architecture [19]

2.4 One-Dimensional Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are widely used for processing two-dimensional data, particularly in image-related tasks. The typical architecture of a CNN comprises an input layer, one or more hidden layers, and an output layer, where the hidden layers commonly include convolutional, pooling, and fully connected layers [21, 22]. The CNNs can also be defined as 1D-CNN to handle Time-Series data. The key difference is the dimensionality of the input data and how the kernel slides across the data [23]. In the case of 1D-CNN applied to time-series data, the convolutional layers slide a kernel (also called a filter) along the time axis of the sequence. This operation allows the network to detect and capture local patterns, trends, or relationships that occur in the data over short periods of time.

Figure 2.2 shows the 1D-CNN structure, where the input layer takes time-series data as the input. The convolutional layers are composed of multiple kernels, where each convolution layer consists of several convolution kernels of the same size, performing convolution operations to generate the corresponding 1D feature maps. The following pooling layer will perform the average pooling [24] or max pooling [25] operation and then send the output to the fully connected layer for final predictions [26].

Assume that given a $1 * 6$ input vector (x_1, x_2, ..., x_6) and the kernel size is 3, which means that the convolution window has a width of 3. During the convolution

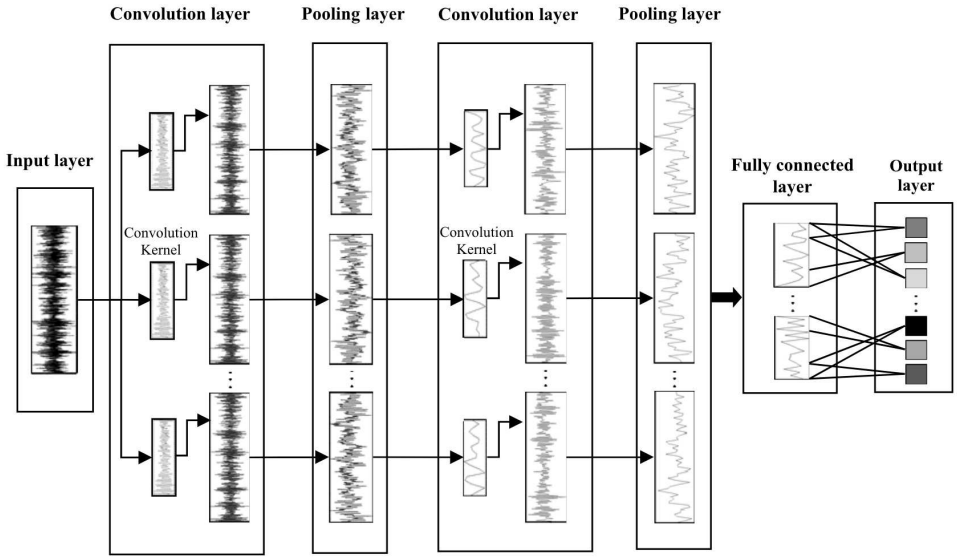


Figure 2.2: The structure of 1D-CNN from [27]

process, the values in the input vector are multiplied by the weights in the kernel window (w_1, w_2, w_3). These products are then summed up to generate the value for the corresponding feature map. In this case, when applying a $1 * 3$ kernel on a $1 * 6$ input vector, the resulting feature vector (y_1, y_2, y_3, y_4) will have a size of $1*4$. Take y_2 as an example, it is obtained by: $y_2 = (w_1 * x_2 + w_2 * x_3 + w_3 * x_4)$. The output from the convolution layer becomes the input for the subsequent layer, allowing the network to progress and extract high-level features as the data flows through the network. After the convolution operation, the typical activation functions such as sigmoid [28], tanh [29], and ReLU [30], will be applied to introduce nonlinearities to the CNNs. The formula for a 1D convolution layer l can be demonstrated by Equation 2.2 [22]:

$$x_j{}^l = f\left(\sum_{j=1}^{M} x_i{}^{l-1} \circledast k_i j^l + b_j{}^l\right) \tag{2.2}$$

where k is the number of convolution kernels, j denotes the kernel sizes, and M represents the channel number of input. Kernel bias is b, f is the activation function and ($\circledast$) is the convolution operator.

Pooling layers are commonly used after the convolutional layers to down-sample the feature maps and reduce the number of parameters while maintaining the major

features. The downsampling helps reduce the complexity and computation load in the subsequent layers [27]. The fully connected layers, also known as dense layers, take all neurons in the feature map output from the previous layer [31, 32]. For a prediction task, the learned features are flattened into a single long vector and passed through fully connected layers before reaching the output layer for final predictions.

Chapter 3

Related Work

Extensive research work has been conducted in the field of power consumption estimation. In this Chapter, we give a brief description of the existing approaches, as well as their limitations. Then we briefly compare the existing approaches with our work and explain the benefits of our work.

3.1 Analytical-based Method

The analytical-based works typically use mathematical models or equations [9, 33] to estimate the power dissipation of the circuits based on the physical capacitance and fundamental electrical principles. The work proposed by Liu and Svensson [34] formulated the average power consumption of each design entity by multiplying the number of gate equivalents with the power consumed by each gate. Some other works [35, 36] related the power consumption of a functional block to the amount of computational effort it performs. The study by [37] introduced a power estimation technique based on the complexity of a Boolean network representation of the design, which exploits the observation that power consumption is proportional to the product of physical capacitance and activity.

Analytical-based methods have the advantage of providing fast predictions with very little input information. Such methods were primarily utilized in the early stages of the development of power estimation techniques. However, as designs become more complex, the limited accuracy of analytical models becomes more apparent, making them less suitable for highly complex VLSI circuits. In addition, analytical models typically assume ideal conditions and dependencies to formulate power consumption,

which may not hold true in complex circuits.

In our work, we provide a power model by capturing and learning the relationship between power consumption and switching activities, i.e., the factors that dominantly affect power consumption. Comparatively, our proposed approach is more applicable to today's complex VLSI circuits.

3.2 Regression-based Method

Numerous earlier works have primarily focused on the regression-based approach. Many studies constructed predictive models using linear regression techniques and applied them to various domains. The model abstraction approach proposed by J. Yang et al. [16] abstracted the relationship between the register toggling profile and power waveform, and enables real-time power estimation by measuring the toggling profile at the register level. The study by D.Kim et al. [38] presented a regression-based signal model and employed the cluster method to pinpoint the primary power dissipation signals. The paper [39] described a general method for developing and optimizing `RTL` power models using linear regression techniques. According to the study by J.Anderson and F.Najm [40], a regression-based prediction model was built in field-programmable gate array (FPGA) designs for the purpose of early-power estimation and power-aware synthesis and layout. Another research [41] worked on deploying a set of acceleration techniques based on regression models to enhance the efficiency specification of `RTL` power estimation tools.

The regression-based techniques at the architectural level have also been the subject of several studies. For example, J. Anderson et al. [40] proposed a methodology for predicting processor power at the pipeline or instruction level, focusing on simulation implementation, sampling, and fitting generalized regression equations. Several works on the architecture level typically use microprocessor performance counters to construct the power model [13, 15].

According to the research by A.K.A.Kumar et al. [42], such simple linear regression techniques at the architectural level suffer from the problem of not capturing relationships well. In general, the current regression-based methods mainly rely on simplistic linear models, which might fall short of capturing intricate dependencies between inputs and power consumption. Consequently, such methods may lead to inaccurate estimates under varying conditions, especially when dealing with nonlinear relationships within complex systems.

In contrast, our ML-based power consumption model excels in capturing both

linear and nonlinear relationships. It could lead to more optimal estimation accuracy because it accounts for complex dependencies that the conventional linear regression methods struggle to capture.

3.3 Advanced ML-based Method

Advanced ML approaches have been explored for power consumption estimation in recent years. Dhotre et al. [7] introduced a methodology for predicting test power, utilizing various supervised learning algorithms. These algorithms, including Linear Least-Square (LLS), K-Nearest Neighbors (KNN), and MultiLayer Perceptron (MLP), were compared within the scope of this study.

The work proposed by Zhou et al. [17] introduced the `PRIMAL` framework which constructs a CNN model through 2D image encoding techniques. However, while achieving relatively high prediction accuracy, this method suffered from reduced efficiency and required a substantial amount of training samples, consequently leading to extended training times. Another work by Zhang et al. [43] presented a Graph Neural Network(GNN) model `GRANNITE`, which was trained using gate-level netlists and corresponding switching rates. One advantage of this model is its adaptability to new circuit designs without the need for retraining. However, it was primarily limited to predicting average power consumption.

In our work, we present a 1D-CNN power consumption model that supports time-based power consumption estimation. This model is accurate down to the cycle level. In contrast to average power consumption estimation methods, our proposed model empowers designers to explore the dynamic changes in power consumption over time. Overall, our approach utilizes advanced ML techniques to ensure accurate and cycle-by-cycle power consumption estimation within a reasonable training time.

Chapter 4

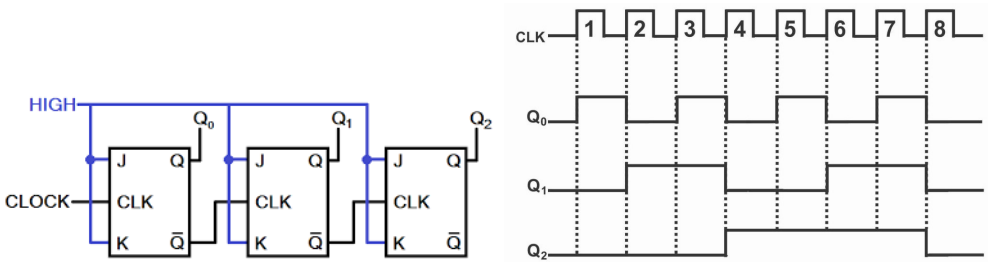
Methodology

This chapter presents our methodology to estimate circuit power consumption based on the RTL circuit switching activity. The proposed framework provides an overview of the entire workflow, generally including three phases: RTL circuit switching activity analysis and power consumption analysis, modeling/training phase, and prediction phase. The details of each phase will be discussed in the following sections.

4.1 Framework

The workflow begins with time-based switching activity and power consumption analysis, as shown in Figure 4.2. Our time-based approach involves a cycle-by-cycle analysis of the circuit switching activity and power consumption, which facilitates building a cycle-accurate power consumption estimation model.

As we introduced in Section 2.2, the RTL circuit's state transitions serve as the input feature, representing the primary contributors to the total power consumption. We used the RTL circuit switching activity matrix to represent the state transitions of a circuit. Figure 4.1(a) gives a simple circuit example to explain how the state transitions of a circuit are related to the switching activity matrix. The circuit is a 3-bit asynchronous binary counter using positive edge triggered JK flip flop. The timing diagram in Figure 4.1(b) shows the transitions of the signals on every positive edge of the clock.



(a) (b)

Figure 4.1: A three-bit asynchronous binary counter circuit with the timing diagram

$$\begin{array}{c} Clock\,Cycle \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} Q0 \\ \left[\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}\right. \end{array} \begin{array}{c} Q1 \\ \begin{array}{c} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{array} \end{array} \begin{array}{c} Q2 \\ \left.\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{array}\right] \end{array} \tag{4.1}$$

According to the timing diagram, logic state transitions are identified, and switching activities are counted for each signal during each clock cycle. Equation 4.1 shows the resulting switching activity matrix, where the columns correspond to signals (Q_0, Q_1, Q_2), and the rows correspond to each clock cycle.

In the first stage of the proposed framework, the time-based Stimulus Database (SDB) is derived through `RTL` simulations. It is the dump of signal activity data from the simulation of the `RTL` design, representing the circuit's transition states and serving as a key ingredient for power analysis. The synthesis database contains gate-level netlists after logic synthesis in the VLSI design flow, as described in Section 2.1. Afterward, the SDB and synthesis database are fed into the power analysis tool to generate the Power Database (PDB). The power profile stored in the PDB records the power consumed by the switching activity for each clock cycle.

The SDB is dumped to a switching activity matrix $\mathbf{X}$, elaborated in Section 4.2.1, while the power consumption profile is represented as a power consumption vector $\mathbf{P}$,

as explained in Section 4.2.2.

The second modeling/training phase takes **X** as the input dataset and **P** as the output dataset to train a convolutional neural network model. The model is able to learn and abstract the relationship between switching activity and corresponding power consumption. Finally, the trained power model is evaluated using testing datasets or applied to a new target input dataset for prediction.

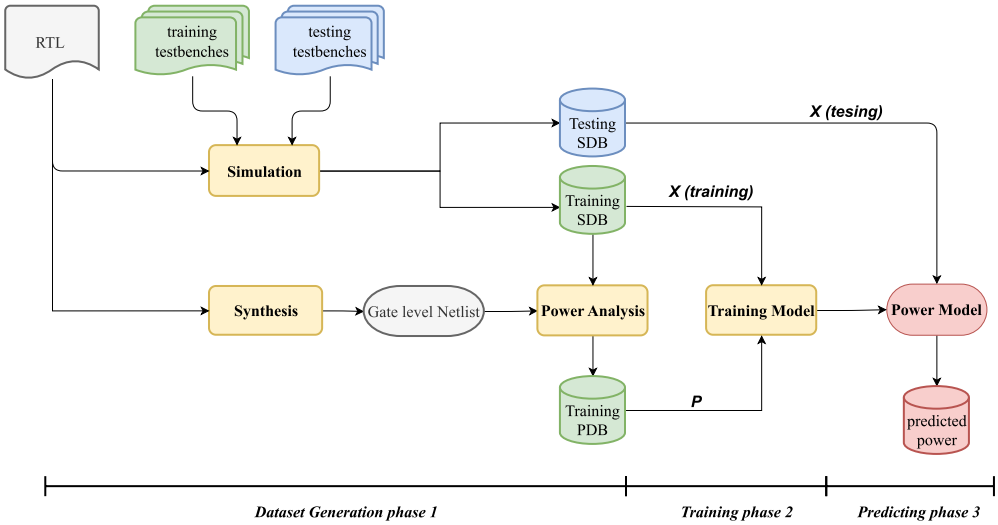


Figure 4.2: Power consumption modeling and estimation workflow

4.2 Dataset Generation (Phase 1)

We split the testbenches used to design the circuit into two categories for generating training and testing datasets, respectively. The switching activity analysis and power analysis are performed in time-based mode, i.e., obtain the toggles and power consumption for each clock cycle. The results of the switching activity analysis will be processed as the input dataset, and power analysis results will be processed as the corresponding output dataset.

4.2.1 Switching Activity Analysis and Reporting

The circuit simulations are performed with XCELIUM [44], a simulation tool in the Cadence suite. The SDB is generated after the simulation of each testbench on the

RTL design. The stimulus comes in various formats that represent switching activity, which quantifies the transitions in the states of registers and Input/Output signals over a certain duration within a circuit.

In this work, two kinds of formats are generated for different intentions, using the `dumptcf` and `dumpvcd` command in XCELIUM. The Value Change Dump (VCD) captures the change of signal values over time, which can be read in the time-based mode for power analysis. The Toggle Count Format (TCF) represents the average switching activity over a certain duration.

In this section, we first discuss the TCF format, which is used to generate the input dataset. To analyze switching behavior on a per-cycle basis, the simulation time is segmented into frames, with each frame corresponding to one clock cycle. The toggle count within each frame is captured and recorded, which indicates how often the pin or net switches between the High Voltage Level (V_{high}) and Low Voltage Level (V_{low}) states.

Therefore, the SDB for each simulation stores multiple TCF files, representing the switching activity for each clock cycle. Then we parsed the TCF file and converted it into a matrix using the *Panda* [45] library in Python. The rows of the matrix represent each consecutive clock cycle, while each column represents a specific signal inside the circuit.

4.2.2 Power Analysis and Reporting

The power analysis is implemented in the Cadence Power Solution tool JOULES [46], which is integrated with Genus [47], a logic synthesis tool in Cadence. We will discuss the workflow step by step.

1. **Read RTL Design**: The process begins by reading the RTL design, libraries, and other technical files for elaboration.

2. **Read Stimulus Database**: The stimulus file will then be read into JOULES. Since we aim to compute the power consumption mainly caused by the switching activity for each cycle, the stimulus containing the switching activity information should be read in time-based mode. The TCF database can not be read in this mode because it only records average activity statistics. Moreover, although the simulation has been segmented to multiple stimuli to represent the switching activity for each clock cycle, as we discussed previously in Section 4.2.1, the Joules integrated with Genus does not support the feature to read multiple stimuli yet.

Therefore, the `VCD` database is read into JOULES and the stimulus information is saved as frames in JOULES SDB. Frames equal to one clock cycle that are extracted from the `VCD` stimulus.

3. **Synthesis**: JOULES offers the command `power_map` in this step to perform synthesis and create a fully functional prototype netlist for power analysis. The prototype netlist database is saved for use in the next power analysis step.

4. **Annotation**: For accurate power analysis, the `rtlstim2gate` feature in JOULES is used to map RTL stimulus on gate-level netlist with high annotation.

5. **Power analysis & Report**: After reading the netlist DB and the Joules SDB into Joules, power consumption is computed in the time-based mode and reported by frame.

Consequently, the cycle-accurate power profile over the simulation time can be represented as a vector, where each element represents the power consumption value per clock cycle.

4.2.3 Summary

In summary, the generated dataset comprises pairs of switching activity matrix $\mathbf{X}$ and power consumption vector $\mathbf{P}$ for each simulation on a certain testbench, as shown in Equation 4.2. To be specific, the dataset includes the following components:

- A time series switching activity matrix $\mathbf{X}$
  Each row corresponds to one clock cycle, representing a specific time interval. Within each clock cycle, the toggle count numbers for each signal are stored. x_{ij} represents the toggle counts for signal j during cycle i. The number of rows in the matrix depends on the total number of clock cycles, while the number of columns depends on the total number of selected signals. The matrix will be used as an input dataset for training the model.

- A time series power consumption vector $\mathbf{P}$
  Each row represents one clock cycle. The vector stores power consumption values within each time interval. The number of rows in the vector depends on the total number of clock cycles for the specific testbench. The output dataset serves as the target output for the prediction of the model.

$$\begin{array}{c} \\ cycle\,1 \\ cycle\,2 \\ \vdots \\ cycle\,m \end{array} \overbrace{\begin{array}{c} signal\,1 \\ \left[\begin{array}{c} x_{11} \\ x_{21} \\ \vdots \\ x_{m1} \end{array}\right. \end{array} \begin{array}{c} signal\,2 \\ \begin{array}{c} x_{12} \\ x_{22} \\ \vdots \\ x_{m2} \end{array} \end{array} \begin{array}{c} \dots \\ \begin{array}{c} \dots \\ \dots \\ \ddots \\ \dots \end{array} \end{array} \begin{array}{c} signal\,n \\ \left.\begin{array}{c} x_{1n} \\ x_{2n} \\ \vdots \\ x_{mn} \end{array}\right] \end{array}}^{\mathbf{X}} \rightarrow \overbrace{\begin{array}{c} power \\ \left[\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_m \end{array}\right] \end{array}}^{\mathbf{P}} \tag{4.2}$$

4.3 Power Modeling with 1D-CNN (Phase 2 & 3)

Given the obtained time-series datasets **X** and **P**, explained in Section 4.2.3, we propose using 1D-CNN for power modeling. The implementation of the 1D-CNN model was carried out in Python, utilizing the *Keras API* [48] and *scikit-learn* [49] for data processing. The aforementioned datasets were split into training datasets (80%) and testing datasets (20%).

In order to maintain the time sequential information in the input dataset and output dataset, the datasets were chunked into a three-dimensional array with dimensions (chunks, frames, features). Each chunk contains a sequence of consecutive clock cycles, which represents the switching activity and power consumption for all signals during a specific time window. The 'chunks' is the number of chunks into which the input and output datasets are divided. The 'frames' means the length of the sequence, i.e., the number of consecutive clock cycles that make up each chunk in the datasets. For the input activity dataset, the number of features corresponds to the number of signals. The output power dataset consists of only one feature per frame, representing the power consumption value for that specific time interval.

Figure 4.3 illustrates the chunking process with a simple example. Given an input dataset of size $(10^4 \times 28)$ and a corresponding output dataset of size $(10^4 \times 1)$. Assuming a chunk size of 10, the total number of chunks becomes 10^3. As a result, the input dataset is partitioned into chunks, giving the chunked input dataset a shape of $(10^3 \times 10 \times 28)$. The chunked output dataset is in the shape of $(10^3 \times 10 \times 1)$. The model learns to map each chunk of the input dataset to the corresponding chunk of the output dataset by analyzing the patterns and correlations within each chunk.

To ensure comparable features for model training and evaluation, it is necessary to process and transform the dataset into a normalized range before passing it into the model. We used the *MinMaxScaler* from the *sklearn* library [49] to rescale the

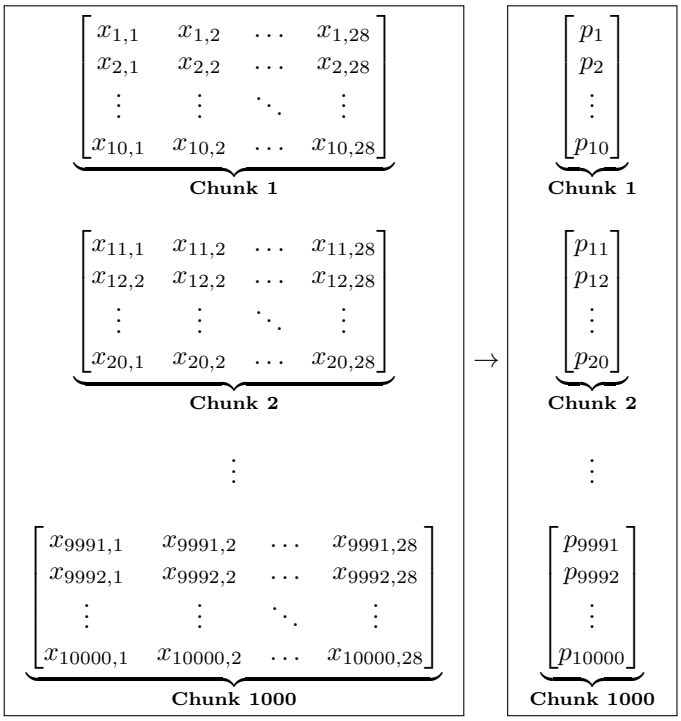


Figure 4.3: A simple example of chunked dataset structure. Each chunked matrix **Chunk i** corresponds to a subset of the dataset.

values of the dataset to a specified range between 0 and 1 as shown in Equation 4.3. The normalization process plays a crucial role in improving the model's convergence efficiency during the training phase. Additionally, it preserves the relative relationships between data points and maintains the patterns present in the original distribution.

$$x_{\text{scaled}} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{4.3}$$

where x_{scaled} represents the scaled value of the input data x_i. $x_{\max}$ represents the maximum value of the dataset. $x_{\min}$ represents the minimum value of the dataset.

The overall structure of the 1D-CNN follows the architecture described in Chapter 2.4, where the input flows through the convolutional layers, pooling layers, flattened layers, and fully connected layers. We used *KerasTuner* [50] to fine-tune the 1D-CNN model. The fine-tuned hyper-parameters include the number of convolutional filters, the number of convolutional layers and dense layers, the units per layer, and the learning rate.

The adoption of a kernel size of 3 in this thesis was carefully deliberated based on a balance between capturing local patterns and minimizing computational complexity. 'ReLU' activation function is used for each convolutional and dense layer to introduce non-linearity to the network, enabling the complex relationships between the input data and the learned features can be modeled. We have explored different configurations on the hyper-parameters, varying the number of convolutional layers and dense layers from 2 to 5. To downsample the spatial dimensions of feature maps, each convolutional layer is followed by a 1D max pooling layer with a pooling size of 2 and stride size of 1. Additionally, we tuned the number of units for each dense layer and the number of filters for each Conv1D layer, setting the search range between 4 and 256 with a step of 4. The learning rate was fine-tuned using three possible values: 1e-2, 1e-3, and 1e-4.

The model is compiled with the Adam optimization algorithm [51] to minimize the loss function root mean squared error (RMSE) shown in Equation 4.4, which measures the difference between the predicted power values using the model, and the reference power values in the output dataset $\mathbf{P}$.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - \hat{p}_i)^2} \tag{4.4}$$

where N is the number of data points, p_i is the reference power value at clock cycle i, $\hat{p}_i$ is the predicted power value at cycle i.

The hyperparameter tuning process was conducted using the *RandomSearch* algorithm, which executed 50 trials to search for the optimal configuration. Each trial consisted of 20 epochs of training and utilized a batch size of 32, ensuring a balanced trade-off between computational efficiency and model optimization.

Chapter 5

Experimental Results

In this chapter, we present the experiments and results during the evaluation of our proposed methodology. The experimental setup used a simple digital circuit design to initially build and configure our framework, i.e., the dataset generation, dataset processing, and 1D-CNN hyperparameter configurations. To further validate our overall methodology, we proceeded by employing our framework on a complex and realistic cryptographic circuit, thereby examining its effectiveness in real practical application scenarios.

5.1 Metric

The NRMSE has been used as the evaluation metric in the experiments. NRMSE provides a normalized measure of the model's power consumption prediction error, ensuring that the evaluation metric is independent of the scale of the data. The predictive capability of the developed power consumption model has been evaluated by quantifying the difference between predicted and reference power values using Equation 5.1.

$$NRMSE = \frac{1}{\Delta p} \times \sqrt{\frac{1}{N}\sum_{i=1}^{n}(p_i - \hat{p}_i)^2} \times 100\% \tag{5.1}$$

where N represents the number of clock cycles, i is the index for each cycle. p_i represents the reference power value at cycle i, $\hat{p}_i$ represents the predicted power value at cycle i. Δp represents the difference of the reference power vector p, i.e. $(p_{max} - p_{min})$. Equation 5.1 calculates the square root of the average of the squared

differences between the predicted $\hat{p}_i$ and reference power values p_i, and then normalizes it by the range of the reference power values (Δp).

5.2 Experimental Setup

The Serial Peripheral Interface slave-to-memory master circuit [52] has been chosen in the experimental setup phase, which is commonly utilized for inter-chip communication in various electronic systems. This circuit includes a total of 28 signals, comprising both input/output ports and internal signals in this simple case. The signals are designed to have a bit-width of 4. It serves as a simple case for the initial configuration and evaluation of our proposed framework. The simulation time of this circuit is 10^7 nanoseconds (ns) with a clock period of 10 ns, thereby leading to a total of 10^6 clock cycles.

5.2.1 Datasets Generation

The process of generating the input dataset for our model begins with dividing the simulation time into multiple frames, and the frame length is aligned with the clock period, as we aim to obtain the circuit switching activity cycle-by-cycle. Therefore, in this example, the simulation time is divided into 10^6 frames, each spanning a duration of 10 ns. The switching activity results are represented by multiple `TCF` files after simulation.

Consequently, each generated `TCF` records the switching activity for one clock cycle. Combining all the `TCF` files, we obtained a switching activity matrix of dimensions ($10^6 \times 28$), where each row represents one frame, and the columns correspond to the 28 signals.

After generating the `VCD` database, the power consumption vector $\mathbf{P}$ in the output dataset has been derived as described in Section 4.2.2. The power computation analysis has been performed in time-based mode, which means the power consumption values were calculated for each clock cycle. As a result, the power consumption vector has a size of ($10^6 \times 1$), capturing the power consumption value consumed by the switching activity for each frame (clock cycle) during the simulation.

5.2.2 Power Modeling

As discussed in Section 4.3, after obtaining the input and output datasets, it is essential to divide the datasets into optimally sized chunks. If the chunk size is too small,

the time-series dataset could become fragmented. In this case, the model might struggle to capture the sequential trends in switching activity and corresponding power consumption over several consecutive cycles. As a result, the model's capability to learn from the tiny chunks is limited. On the other hand, if the chunk size is too large, there might be an insufficient amount of training data available for the model. We examined the impact of different chunk sizes on the model performance and general capability for handling data. We finally decided to partition the input dataset and output dataset into 10^6 chunks in this example, while each chunk consists of 100 frames (consecutive clock cycles). For the input dataset, the reshaped switching activity matrix size is $(10^6 \times 100 \times 28)$. The size of the reshaped output power vector is $(10^6 \times 100 \times 1)$.

In this example, the trained 1D-CNN model consists of 3 convolutional layers with varying numbers of filters: 132, 120, and 152, followed by max-pooling layers. It also includes 2 dense layers with units 156 and 236, contributing to pattern recognition. The final dense layer has 100 units to match the output size of each chunk. The model employs a learning rate of 0.001 for optimization.

5.2.3 Evaluation & Results Analysis

The trained model has been used to predict the power consumption using the switching activity matrix from the testing dataset. The predicted power consumption, obtained by the model, and the reference power consumption from the testing dataset were compared using the NRMSE in Equation 5.1, which achieved approximately 14% error in this experiment. Figure 5.1 plots the predicted power consumption profile (orange curve) and reference power consumption profile (blue curve) over 200 cycles. The x-axis of the plot corresponds to the number of cycles, while the y-axis represents the power consumption in microwatts. It can be seen that the model can achieve high accuracy in predicting lower power values but struggles in estimating higher power values.

Since the dataset is severely unbalanced, with the majority of data falling within the 0-15 mW range of power consumption values, the model's training process was biased towards learning patterns within this dominant range. The tendency of prediction results is consistent with the observations: the values predicted by the model are distributed between 0 and $15mW$. The model may not have effectively captured the patterns and characteristics associated with power values between 30-35 mW.

Additionally, the limited variation in the reference power profile has a negative

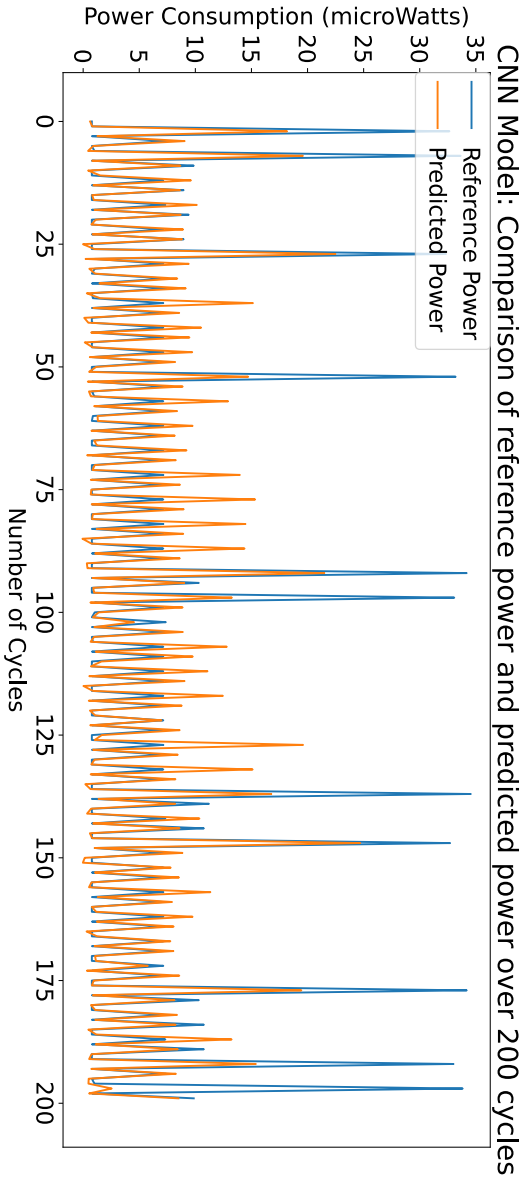


Figure 5.1: Comparison of reference power and predicted power consumption over 200 cycles

impact on the generalization ability of the model. Although the datasets include a sufficient number of samples, the lack of diverse power consumption patterns within the reference data might have hindered the model's capability to learn and generalize to different scenarios.

It is worth noting that the datasets are randomly split into training and testing datasets, which means that 80% of the chunks in the original datasets are randomly selected to form the training dataset, while the remaining 20% of chunks form the testing dataset. Due to the random selection of chunks, the testing dataset is made up of discontinuous chunks, making it not very relevant to make strong conclusions about the power consumption of the original SPI circuit.

As a result, the randomly formed switching activity matrix used for testing does not maintain the time dependencies present in the original input dataset. Thus, the power estimation based on this matrix merely mirrors the learning capacity of the model, lacking any connection to the cycle-by-cycle power consumption of the SPI circuit. In practice, it is more relevant to estimate power consumption or evaluate power models based on different circuit workloads as determined by a user-specified test bench. Therefore, we decided to use the Xoodyak LWC core with over 500 different workloads (testbenches) for our second experiment.

5.3 Xoodyak LWC Core

For further evaluation of our methodology and framework, we have chosen the Xoodyak LWC core described in Section 2.3 as a representative realistic circuit. Based on the hardware architecture of the LWC core described in Section 2.3, the widths of the data buses for the three interface data ports (`PDI,SDI,DO`) were set to 32-bit in this experiment. The total number of monitored signals was reduced to 208 crucial signals to perform switching activity analysis and power consumption analysis more efficiently. The key signals include the input signals within the PDI and SDI interface ports and the internal signals between each block shown in Figure 2.1.

Therefore, the performance of the framework is evaluated on a more complex circuit design. The complexity of this circuit allows us to capture the switching activity and the corresponding power consumption patterns with more variations. By utilizing this LWC core for further evaluation, we aim to assess the effectiveness and scalability of our proposed framework in handling real-world circuit scenarios.

5.3.1 Dataset Generation

We used the KAT test data from [19], comprising 535 test data vectors with different AD, Plaintext, Ciphertext, and Hash Message lengths. The hardware circuit testbench (`LWC_TB`) was configured with these different test data vectors as input, which therefore have varying simulation times. To generate the training dataset, 80% (429) of the test data vectors were randomly selected, while the remaining 20% (106) were used for the testing dataset. For each testbench, the switching activity analysis and power consumption analysis were conducted following the workflow described in Section 4.1.

The training and testing datasets were chunked as explained in Section 4.3. Given that each testbench has a duration of clock cycles varying from 58 to 150, we chose a chunk size of 10 clock cycles. In cases where the dataset size is not evenly divisible by the chosen chunk size of 10, zero padding was applied to accommodate the remaining data points.

5.3.2 Power Modeling

The training datasets consisting of 429 pairs of switching activity matrices (input datasets) and power consumption vectors (output datasets), were used to train the model. The fine-tuned training on the 1D-CNN model followed the process in Section 4.3, which intends to search for the optimal hyperparameters that would minimize the loss function (RMSE) in Equation 4.4 of the model on the training dataset.

By considering the components including data splitting, optimizer selection, activation function choice, loss function definition, evaluation metric specification, and hyperparameter tuning, the 1D-CNN model was fine-tuned to achieve optimal performance on the given dataset. A learning rate of 0.001 is found to be optimal for the Adam optimization algorithms. Other selected hyperparameters for the 1D-CNN model include a configuration of four layers, with the first layer consisting of 64 filters. The subsequent layers comprise 140, 156, and 116 filters, respectively, contributing to feature extraction from the input data. To capture complex patterns and relationships within the data, the model incorporates two dense layers, with the first dense layer containing 244 neurons. The last dense layer has 10 neurons to match the desired output size of each chunk.

5.3.3 Evaluation & Results Analysis

In this experiment, we have a total of 106 testing datasets, each consisting of a switching activity matrix and a reference power vector obtained by Cadence simulation and

power analysis tools according to Section 4.2. The trained model utilized the switching activity matrix as input to predict power consumption. Subsequently, for each test case, the predicted and reference power consumption were compared by calculating the NRMSE in Equation 5.1. Furthermore, we performed a cycle-by-cycle comparison and analysis of the predicted power profile and the reference power profile.

The evaluation revealed an average NRMSE of 11.61% over all 106 test cases. To further analyze the results, we examined the NRMSE of different test cases: The NRMSE results were found to vary in the range from a minimum of 5.68% error to a maximum of 21.49% error. Figure 5.2 illustrates the distribution of NRMSE values. The horizontal axis shows the range of errors measured in percentage, while the vertical axis indicates the count of occurrences for each corresponding error interval.

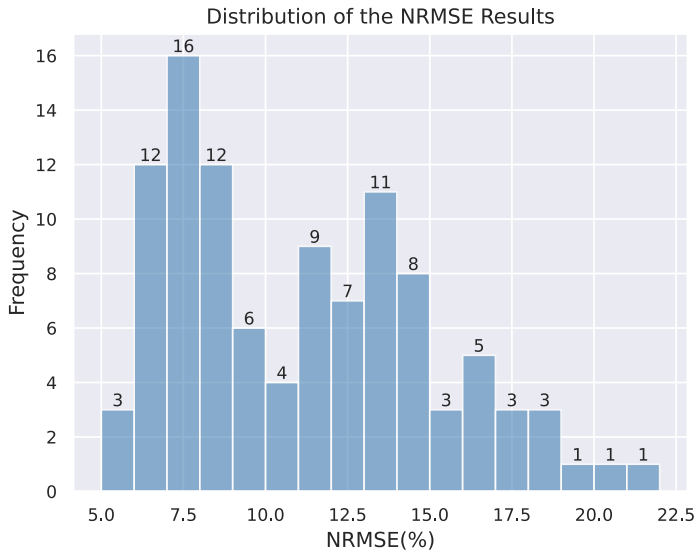


Figure 5.2: The Distribution of NRMSE Results

The results in Figure 5.2 are categorized into three groups based on the NRMSE values, indicating different levels of prediction error (accuracy). The **Best** category indicates a prediction accuracy of 90% and above, i.e., NRMSE $\leq$ 10%, which accounts for 46.67% of the total test cases. The **Average** category comprises NRMSE values between 10% and 15%, which makes up 37.14% of the test cases. The category labeled as **Base** includes NRMSE values between 15% and 25%, accounting for 16.19% of the test cases.

Table 5.1 presents the most representative test cases from the three categories as shown in the 'Category' column: the top three accurate predictions, the three predictions that are close to the median, and the three predictions that rank at the bottom of the accuracy range. The 'Test Case' column gives the identification numbers of the most representative cases in each of the three categories. The third and final columns give the corresponding clock cycles and NRMSE results for each selected test case.

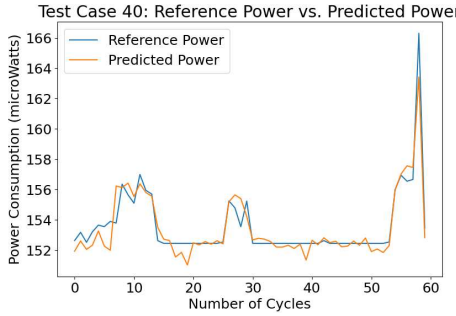
Table 5.1: The NRMSE Results for Three Categories

Category	Test Case	Cycles	NRMSE(%)
Best ($\leq 10\%$)	40	60	5.68
	89	98	5.78
	75	60	5.99
Average (10%-15%)	44	85	10.81
	33	60	10.93
	51	59	11.26
Base ($\geq 15\%$)	13	59	19.84
	57	60	20.07
	16	60	21.49

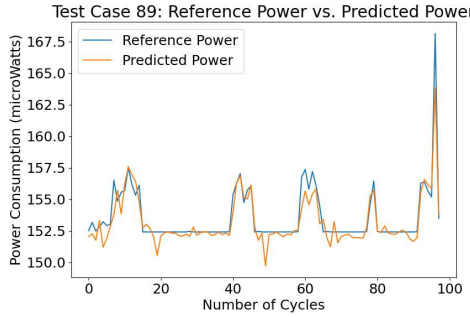
To facilitate a comprehensive analysis, the predicted and reference power profiles for these representative test cases in Table 5.1 are compared cycle by cycle, as shown in Figure 5.3, 5.4, 5.5. For each subfigure, the vertical axis represents the power consumption measured in microwatts, and the horizontal axis corresponds to the number of clock cycles for the simulation of each test bench. The cycle-by-cycle predicted power consumption is depicted by the orange curve, whereas the reference power consumption profile is illustrated by the blue curve.

Figure 5.3 illustrates the top 3 test cases, demonstrating that the model successfully learns and predicts the variations in power consumption patterns. This observation suggests a high prediction accuracy of approximately 95%. The model effectively learns the underlying patterns and features that contribute to changes in power consumption based on the input-switching activity. This positive outcome indicates that the model performs well in capturing the overall trends and variations in power consumption.

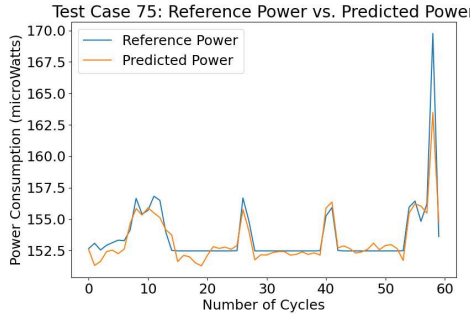
Figure 5.4 shows the three test cases categorized as average. Cycle 24 stands out as an outlier in Figure 5.4(b), meaning that it significantly deviates from the



(a) Test Case 40, NRMSE:5.68%.



(b) Test Case 89, NRMSE:5.78%.

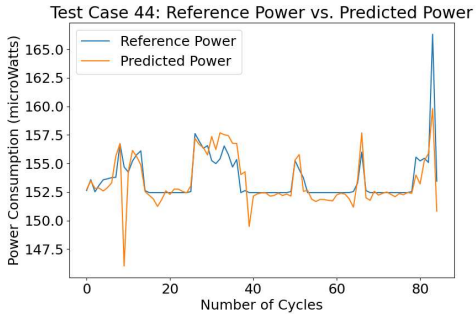


(c) Test Case 75, NRMSE:5.99%.

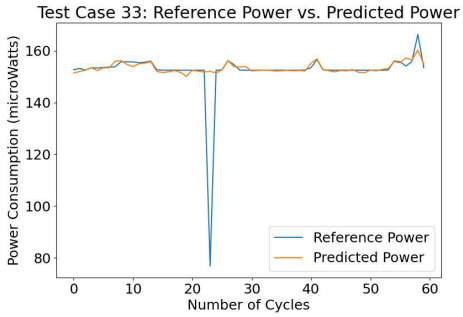
Figure 5.3: Reference power vs. Predicted power for the top 3 cases categorized as **Best**

typical distribution in the reference power datasets. Since the model might not have encountered similar patterns during training, it struggles to generalize to accurately estimate the power of that cycle. The poor ability to predict power consumption accurately for cycles with outliers points out the limitation of the model in handling unforeseen and atypical scenarios.

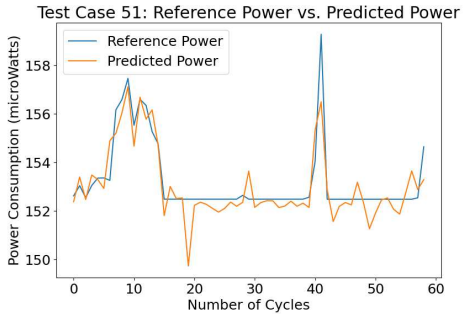
Figure 5.5 showcases the three base cases. The predicted power profile generally aligns with the reference power profile, except for a few cycles. It is noted that the predicted power profile exhibits glitches in cycles where the reference power profile shows no or small variations. This is probably due to the problem of potential overfitting. Overfitting occurs when the model becomes too closely attuned to the training data, hindering the performance of unseen data. As a consequence, the model may introduce inaccuracies when predicting power consumption in cycles without variations.



(a) Test Case 44, NRMSE:10.81%.

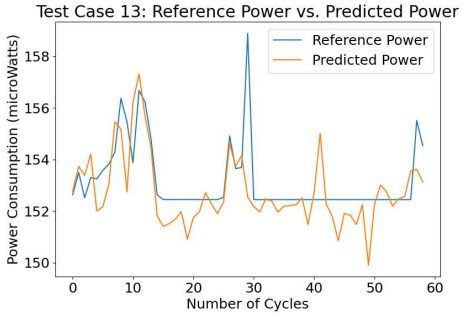


(b) Test Case 33, NRMSE:10.93%.

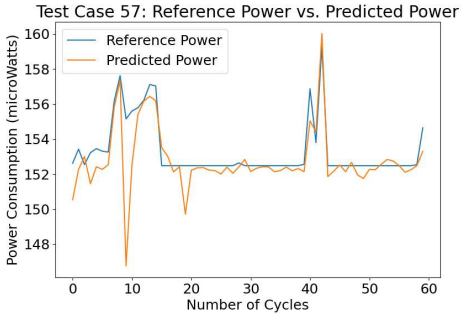


(c) Test Case 51, NRMSE:11.26%.

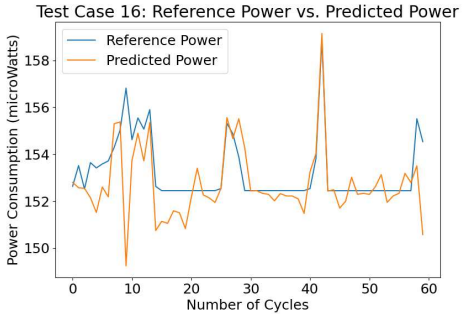
Figure 5.4: Reference power vs. Predicted power for the three cases categorized as **Average**



(a) Test Case 13, NRMSE:19.84%.



(b) Test Case 57, NRMSE:20.07%.



(c) Test Case 16 , NRMSE:21.49%.

Figure 5.5: Reference power vs. Predicted power for the top3 cases categorized as **Base**.

Chapter 6

Discussion

In this chapter, we start with the research question posed at the beginning in Chapter 1 and discuss how the proposed method addresses this question. The experimental results are interpreted and analyzed in more depth. Future research directions are proposed based on the limitations of the current work.

Let us recall the research question raised in Chapter 1: **How can we find the optimal 'sweet spot' for power consumption estimation by achieving a balance between estimation speed and high accuracy?**

It is commonly known that conventional gate-level simulations and power analysis tend to be time-consuming, while high abstraction level power estimation often lacks accuracy. Therefore, to strike a balance between estimation speed and accuracy, our proposed power estimation methodology utilizes the power consumption analysis flow that combines the accuracy nature at low abstraction levels with the efficiency nature at high abstraction levels and introduces ML techniques to it for further acceleration.

We aim to train a ML model capable of learning the relationship between switching activity and power consumption, and then apply the power model to estimate power consumption for various workloads within the same circuit design.

To obtain the training dataset, we followed the power analysis flow with RTL stimulus described in Section 4.2.2. The `RTL` switching activity on the synthesized gate-level netlist is used to perform cycle-by-cycle power calculations. This workflow reduces the time compared to conventional gate-level simulation and power consumption analysis as we introduced in Chapter 1, while still capturing important low-level circuit details, resulting in reasonably accurate power analysis results.

When estimating the power consumption for different workloads, only the switch-

ing activity information from the RTL simulation is required as the input of the trained model in our methodology. This speeds up the power consumption analysis flow compared to commercial power analysis tools, as it eliminates the need for logic synthesis and time-consuming power computation in JOULES.

In our experiments, we applied our power estimation framework to the LWC core using a total of 535 testbenches. Of these, 426 testbenches were selected for training the model, which is referred to as training testbenches (T_{tb}). The remaining 106 testbenches were set aside for testing and prediction, and are known as prediction testbenches (P_{tb}).

Table 6.1 shows the runtime of the power analysis flow using only commercial tools in Section 4.2.2 and the same flow using our proposed power modeling approach as an extension.

The 'Dataset Generation' column provides the runtime for power consumption analysis on T_{tb}. The runtime is the same for both methods because in our proposed power modeling flow, the generation of the training dataset follows the JOULES power analysis flow as shown in Phase 1 in Figure 4.2.

The third column shows the training timing of our proposed power modeling flow, which takes 2.5 hours. In the 'Power Estimation' column, we present the runtime for power consumption analysis on P_{tb} using the JOULES power analysis flow, alongside the runtime for power consumption prediction using the trained 1D-CNN model.

The last column indicates the total runtime for P_{tb}. For the JOULES workflow, power consumption estimation only involves Phase 1 in Figure 4.2, which utilizes the testing testbenches as input. Our proposed power modeling flow necessitates progressing through all phases as shown in Figure 4.2 to perform power consumption estimation for P_{tb}, which includes the generation time of the training dataset, model training time, and prediction time for P_{tb}.

	Dataset Collection (T_{tb})	Model Training	Power Estimation (P_{tb})	Total Runtime on P_{tb}
JOULES Power Analysis Flow with RTL Stimulus	1 hour	-	**10 mins**	10 mins
Proposed Power Analysis flow with 1D-CNN Model	1 hour	2.5 hours	2 mins	**212 mins**

Table 6.1: Comparison of the runtime of the Power Analysis Flow using JOULES and Our Proposed Power Analysis flow with 1D-CNN Model

It is noted that the total runtime on P_{tb} in Table 6.1 can not reveal the efficiency of our current model. This is attributed to the limited count of 106 testbenches used for testing purposes and the relatively short simulation periods for each testbench. Consequently, it did not take too much time for power consumption analysis using JOULES. However, the measure of efficiency will become evident when performing power consumption estimation for a large number of input workloads under the same design.

For instance, in the context of power-based side-channel attacks, it could involve thousands or even millions of different testbenches, each simulating a potential attack scenario. Considering a scenario with 10^4 testbenches, the power estimation duration using JOULES is expected to extend to approximately 10^3 minutes in this case. In contrast, our proposed power modeling flow exclusively necessitates prediction and omits time-consuming Phase 1 and Phase 2. Therefore the runtime is notably more efficient, estimated to be around 300-400 minutes. This comparison illustrates the efficiency of our approach in streamlining the power consumption estimation process for large workloads.

The experimental results in Chapter 5 show that the power estimation accuracy varies across test cases, with most cases exhibiting relatively low estimation errors (5-10%), while a few cases exhibit some deviations from the reference power consumption values (20% error).

To enhance the proposed methodology and improve the performance of the power estimation model, we propose some potential research directions for future work:

- *Handling Outliers:*
  The results in Section 5.3.3 show that it can be challenging to predict the cycles with outliers in the dataset. Therefore, to improve the accuracy of power consumption predictions for those cycles with outliers, it is essential to refine the training dataset. Careful curation of the dataset should adequately represent a diverse range of power consumption patterns, including rare or unusual cases. This will enable the model to learn and generalize better, even for instances that deviate significantly from the typical pattern.

- *Handling Steady States:*
  The results also point out the problems for those steady-state cycles, where we refer to some cycles in the reference power profiles with little or no variation. We consider introducing regularization techniques during model training to prevent overfitting and thus improve the ability of the model to handle cycles with small

power variations.

- *Capturing RTL Activity and Power more Accurately:*
  In our current methodology, the monitored signals of the circuit are manually selected for switching activity analysis and power consumption analysis. The model is subsequently trained based on the relationship between the captured activity and power consumption. However, it is possible that we may overlook certain signals that have a significant influence on power consumption. This limitation can potentially lead to less accurate power estimation results using the current model. An interesting area for future research is to explore how to select the most representative signals for switching activity and power consumption analysis. By capturing the `RTL` switching activity and power consumption more accurately through optimized signal analysis and selection approaches, we can enhance the power estimation model and ensure that it accounts for all relevant and critical signals contributing to the power consumption.

- *Enlarging the Dataset:*
  The current experiment generated relatively small training and testing datasets. Increasing the size and diversity of the dataset may provide more representative samples and improve the learning ability of the model. In addition, including more testbenches for testing would provide a broader range of scenarios to assess the performance of the model and enhance its robustness.

- *Complex Circuit Designs:*
  Currently, our power estimation model is applied to relatively small realistic circuits. We consider applying the model to more complex circuits in future works. The effectiveness and scalability of the model can be further evaluated with larger and more complex circuit designs, as they tend to have more signals, longer runtime for simulation and power analysis, and more intricate power consumption patterns.

- *More Efficient Training for Different Designs:*
  Another noticeable limitation is that the model needs to be retrained for every new circuit design. While the common application scenario is inferring power consumption under different testing workloads on the same circuit design, when it comes to the scenarios that require different designs, the retraining process for each new circuit design can be time-consuming, particularly for larger and more complex circuit designs.

Chapter 7

Conclusions

In this work, we have introduced a power modeling approach that accelerates cycle-accurate power estimation during the RTL design phase.

Our methodology was applied to a lightweight cryptographic core, and the results demonstrated its effectiveness in predicting the power consumption of most test cases. The 1D-CNN model achieved an optimal NRMSE of 5.68% for cycle-by-cycle power estimation.

The developed model can be generalized to different workloads under the same circuit design and greatly release the computational burden of post-synthesis power computation in the Cadence Power Solution tool.

In conclusion, our work provides an agile, flexible design flow that enables efficient and accurate cycle-accurate power estimation in the early stages of hardware design. Moving forward, the framework empowers designers to make informed decisions and timely adjustments on power consumption, which can be generalized to various power-related domains, such as identifying vulnerabilities in power consumption that could be exploited by power analysis side-channel attacks.

Bibliography

[1] Massoud Pedram. "Design Technologies for Low Power VLSI". In: *Encyclopedia of Computer Science and Technology* 36 (1997), pp. 73–96.

[2] Kanika Kaur and Arti Noor. "Strategies & Methodologies for Low Power VLSI Designs: A Review". In: *International Journal of Advances in Engineering & Technology* 1.2 (2011), p. 159.

[3] Gajski and Kuhn. "Guest Editors' Introduction: New VLSI Tools". In: *Computer* 16.12 (Dec. 1983), pp. 11–14.

[4] Andreas Gerstlauer and Daniel D. Gajski. "System-Level Abstraction Semantics". In: *Proceedings of the 15th International Symposium on System Synthesis*. ISSS '02. New York, NY, USA, Oct. 2, 2002, pp. 231–236.

[5] B. Chen and I. Nedelchev. "Power Compiler: A Gate-Level Power Optimization and Synthesis System". In: *Proceedings International Conference on Computer Design VLSI in Computers and Processors*. International Conference on Computer Design VLSI in Computers and Processors. Austin, TX, USA, 1997, pp. 74–79.

[6] Yanqing Zhang. "Deep Learning for Power and Switching Activity Estimation". In: *Machine Learning Applications in Electronic Design Automation*. 2022, pp. 85–114.

[7] Harshad Dhotre et al. "Machine Learning-Based Prediction of Test Power". In: *2019 IEEE European Test Symposium (ETS)*. 2019 IEEE European Test Symposium (ETS). Baden-Baden, Germany, May 2019, pp. 1–6.

[8] H V Ranjitha, Sujatha Hiremath, and Saya Goud Langadi. "RTL Power Estimation: Early Design Cycle Approach for SoC Power Sign-Off". In: *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. 2018 3rd IEEE International Conference on

Recent Trends in Electronics, Information & Communication Technology (RTEICT). Bangalore, India, May 2018, pp. 480–484.

[9] P.E. Landman and J.M. Rabaey. "Activity-Sensitive Architectural Power Analysis". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.6 (June 1996), pp. 571–587.

[10] K.M. Buyuksahin and F.N. Najm. "Early Power Estimation for VLSI Circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.7 (July 2005), pp. 1076–1088.

[11] S. Gupta and F.N. Najm. "Power Modeling for High-Level Power Estimation". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8.1 (), pp. 18–29.

[12] Deming Chen et al. "High-Level Power Estimation and Low-Power Design Space Exploration for FPGAs". In: *2007 Asia and South Pacific Design Automation Conference*. 2007 Asia and South Pacific Design Automation Conference. Yokohama, Japan, Jan. 2007, pp. 529–534.

[13] W. Lloyd Bircher and Lizy K. John. "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events". In: *2007 IEEE International Symposium on Performance Analysis of Systems & Software*. 2007 IEEE International Symposium on Performance Analysis of Systems & Software. San Jose, CA, USA, Apr. 2007, pp. 158–168.

[14] Xinnian Zheng, Lizy K. John, and Andreas Gerstlauer. "LACross: Learning-Based Analytical Cross-Platform Performance and Power Prediction". In: *International Journal of Parallel Programming* 45.6 (Dec. 2017), pp. 1488–1514.

[15] Mark Sagi et al. "A Lightweight Nonlinear Methodology to Accurately Model Multicore Processor Power". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.11 (Nov. 2020), pp. 3152–3164.

[16] Jianlei Yang et al. "Early Stage Real-Time SoC Power Estimation Using RTL Instrumentation". In: *The 20th Asia and South Pacific Design Automation Conference*. 2015 20th Asia and South Pacific Design Automation Conference (ASP-DAC). Chiba, Japan, Jan. 2015, pp. 779–784.

[17] Yuan Zhou et al. "PRIMAL: Power Inference Using Machine Learning". In: *Proceedings of the 56th Annual Design Automation Conference 2019*. DAC '19: The 56th Annual Design Automation Conference 2019. Las Vegas NV USA, 2019, pp. 1–6.

[18] M. Nemani and F.N. Najm. "Towards a High-Level Power Estimation Capability [Digital ICs]". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.6 (June 1996), pp. 588–598.

[19] Jens-Peter Kaps et al. *A Comprehensive Framework for Fair and Efficient Benchmarking of Hardware Implementations of Lightweight Cryptography*. Cryptology ePrint Archive, Paper 2019/1273. https://eprint.iacr.org/2019/1273. 2019. URL: https://eprint.iacr.org/2019/1273.

[20] Joan Daemen et al. "The Design of Xoodoo and Xoofff". In: *IACR Transactions on Symmetric Cryptology* (Dec. 13, 2018), pp. 1–38.

[21] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. "Artificial Neural Networks: A Tutorial". In: *Computer* 29.3 (Mar. 1996), pp. 31–44.

[22] Serkan Kiranyaz et al. *1D Convolutional Neural Networks and Applications: A Survey*. May 9, 2019. arXiv: 1905.03554 [cs, eess]. URL: http://arxiv.org/abs/1905.03554. preprint.

[23] Salim Malek, Farid Melgani, and Yakoub Bazi. "One-Dimensional Convolutional Neural Networks for Spectroscopic Signal Regression: Feature Extraction Based on 1D-CNN Is Proposed and Validated." In: *Journal of Chemometrics* 32.5 (May 2018), e2977.

[24] Tao Wang et al. "End-to-End Text Recognition with Convolutional Neural Networks". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). Nov. 2012, pp. 3304–3308.

[25] Y-Lan Boureau, Jean Ponce, and Yann LeCun. "A Theoretical Analysis of Feature Pooling in Visual Recognition". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Madison, WI, USA, 2010, pp. 111–118.

[26] Jiuxiang Gu, Zhenhua Wang, and Jason Kuen. "Recent Advances in Convolutional Neural Networks". In: *Pattern Recognition* 77 (May 2018), pp. 354–377.

[27] Shuzhan Huang et al. "Signal Status Recognition Based on 1DCNN and Its Feature Extraction Mechanism Analysis". In: *Sensors* 19.9 (9 Jan. 2019), p. 2018.

[28] Jun Han and Claudio Moraga. "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning". In: *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*. IWANN '96. Berlin, Heidelberg, 1995, pp. 195–201.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (7553 May 2015), pp. 436–444.

[30] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Madison, WI, USA, 2010, pp. 807–814.

[31] Geoffrey E. Hinton et al. *Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors*. July 3, 2012. arXiv: 1207.0580 [cs]. preprint.

[32] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Vol. 8689. Cham, 2014, pp. 818–833.

[33] Sheng Li et al. “The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing”. In: *ACM Transactions on Architecture and Code Optimization* 10.1 (Apr. 2013), pp. 1–29.

[34] Dake Liu and C. Svensson. “Power Consumption Estimation in CMOS VLSI Chips”. In: *IEEE Journal of Solid-State Circuits* 29.6 (June 1994), pp. 663–670.

[35] Diana Marculescu, Radu Marculescu, and Massoud Pedram. “Information Theoretic Measures of Energy Consumption at Register Transfer Level”. In: *Proceedings of the 1995 International Symposium on Low Power Design - ISLPED ’95*. The 1995 International Symposium. Dana Point, California, United States, 1995, pp. 81–86.

[36] F.N. Najm. “A Survey of Power Estimation Techniques in VLSI Circuits”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2.4 (Dec. 1994), pp. 446–455.

[37] Farid N. Najm. “Towards a High-Level Power Estimation Capability”. In: *Proceedings of the 1995 International Symposium on Low Power Design - ISLPED ’95*. The 1995 International Symposium. Dana Point, California, United States, 1995, pp. 87–92.

[38] Donggyu Kim et al. “Simmani: Runtime Power Modeling for Arbitrary RTL with Automatic Signal Selection”. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO ’52. New York, NY, USA, 2019, pp. 1050–1062.

[39] Alessandro Bogliolo, Luca Benini, and Giovanni De Micheli. “Regression-Based RTL Power Modeling”. In: *ACM Transactions on Design Automation of Electronic Systems* 5.3 (July 2000), pp. 337–372.

[40] J.H. Anderson and F.N. Najm. “Power Estimation Techniques for FPGAs”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12.10 (Oct. 2004), pp. 1015–1027.

[41] S. Ravi, A. Raghunathan, and S. Chakradhar. “Efficient RTL Power Estimation for Large Designs”. In: *16th International Conference on VLSI Design, 2003. Proceedings.* 16th International Conference on VLSI Design. Concurrently with the 2nd International Conference on Embedded Systems Design. New Delhi, India, 2003, pp. 431–439.

[42] Ajay Krishna Ananda Kumar et al. “Machine Learning-Based Microarchitecture-Level Power Modeling of CPUs”. In: *IEEE Transactions on Computers* 72.4 (Apr. 2023), pp. 941–956.

[43] Yanqing Zhang, Haoxing Ren, and Brucek Khailany. “GRANNITE: Graph Neural Network Inference for Transferable Power Estimation”. In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 2020 57th ACM/IEEE Design Automation Conference (DAC). San Francisco, CA, USA, 2020, pp. 1–6.

[44] *Xcelium Logic Simulation*. Oct. 2022. URL: https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html.

[45] *Pandas - Python Data Analysis Library*. May 2023. URL: https://pandas.pydata.org/.

[46] *Joules RTL Power Solution*. Nov. 2022. URL: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html.

[47] *Genus Synthesis Solution*. Aug. 2022. URL: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.

[48] *Keras Documentation: Keras API Reference*. May 2021. URL: https://keras.io/api/.

[49] *Sklearn.Preprocessing.MinMaxScaler*. scikit-learn. Aug. 2020. URL: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[50] *Keras Documentation: KerasTuner*. June 2020. URL: https://keras.io/keras_tuner/.

[51] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. arXiv: 1412.6980 [cs]. preprint.

[52] . *Serial Peripheral Interface - an Overview — ScienceDirect Topics*. Jan. 2015. URL: https://www.sciencedirect.com/topics/computer-science/serial-peripheral-interface.