# NeuPow: A CAD Methodology for High Level Power Estimation Based on Machine Learning

Yehya Nasser, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, Luigi Raffo

▶ **To cite this version:**

Yehya Nasser, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, et al.. NeuPow: A CAD Methodology for High Level Power Estimation Based on Machine Learning. ACM Transactions on Design Automation of Electronic Systems, 2020, 25 (5), pp.1-29. 10.1145/3388141 . hal-02518770

**HAL Id: hal-02518770**
**https://hal.science/hal-02518770v1**

Submitted on 25 Mar 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# NeuPow: A CAD Methodology for High Level Power Estimation Based on Machine Learning
# Submitted to the Special Issue on Machine Learning for CAD (ML-CAD)

YEHYA NASSER, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, France
CARLO SAU, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Italy
JEAN-CHRISTOPHE PRÉVOTET, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, France
TIZIANA FANNI, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Italy
FRANCESCA PALUMBO, Università di Sassari, IDEA Lab., Italy
MARYLINE HÉLARD, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, France
LUIGI RAFFO, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Italy

In this paper, we present a new, simple, accurate and fast power estimation technique that can be used to explore the power consumption of digital system designs at an early design stage. We exploit the machine learning techniques to aid the designers in exploring the design space of possible architectural solutions, and more specifically, their dynamic power consumption, which is application, technology, frequency and data stimuli dependent. To model the power and the behavior of digital components, we adopt the Artificial Neural Networks (ANNs), while the final target technology is Application Specific Integrated Circuit (ASIC). The main characteristic of the proposed method, called NeuPow, is that it relies on propagating the signals throughout connected ANN models to predict the power consumption of a composite system. Besides a baseline version of the NeuPow methodology that works for a given predefined operating frequency, we also derive an upgraded version that is frequency aware, where the same operating frequency is taken as additional input by the ANN models. To prove the effectiveness of the proposed methodology, we perform different assessments at different levels. Moreover, technology and scalability studies have been conducted, proving the NeuPow robustness in terms of these design parameters. Results show a very good estimation accuracy with less than 9% of relative error independently from the technology and the size/layers of the design. NeuPow is also delivering a speed-up factor of about 84× with respect to the classical power estimation flow.

CCS Concepts: • **ASICs** → **Computer aided design**; • **Machine Learning** → *Artificial Neural Network*; • **Power Consumption** → *Modeling*; • **Power Estimation** → CAD tools.

Additional Key Words and Phrases: power consumption, neural networks, modeling, estimation, methodology

Authors' addresses: Yehya NASSER, yehya.nasser@insa-rennes.fr, yehya.nasser@outlook.com, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, France, Rennes, France, 35000; Carlo Sau, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Cagliari, Italy; Jean-Christophe Prévotet, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, Rennes, France; Tiziana Fanni, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Cagliari, Italy; Francesca Palumbo, Università di Sassari, IDEA Lab. Sassari, Italy; Maryline Hélard, Univ Rennes, INSA Rennes, IETR, CNRS, UMR 6164, Rennes, France; Luigi Raffo, Università di Cagliari, Dept. of Electrical and Electronics Engineering, Cagliari, Italy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
XXXX-XXXX/2020/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

**ACM Reference Format:**
Yehya NASSER, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, and Luigi Raffo. 2020. NeuPow: A CAD Methodology for High Level Power Estimation Based on Machine Learning Submitted to the Special Issue on Machine Learning for CAD (ML-CAD) . 1, 1 (March 2020), 29 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Nowadays, power consumption has become a critical performance metric in a wide range of electronic devices, from autonomous embedded systems based on batteries to more complex systems requiring a high computing power. In a near future, a huge amount of these devices will clearly have an impact on the environment and on the worldwide energy consumption in particular [8]. One attempt to reduce the consumed power of such devices would consist in allowing designer to explore various hardware architectures very soon in the design process. This would enable to reach an optimal architecture in terms of performance and power consumption.

In order to optimize the power consumed in the end product, it is necessary to have an idea about the power profile and to perform a deep analysis of the different factors that may impact power. Generally, it is possible to get an idea about this profile by measuring the device consumption in its real conditions of execution. Measuring power is not always easy to perform and designers must complete the full design flow in order to obtain the first result, which is often prohibitive. Moreover, each minor change in the design imposes to perform a new power evaluation. It may also happen that, after running a full design flow, the evaluated consumed power is not compatible with the initial requirements. In this case, designers have to re-run a full design process from scratch, which is of course time consuming and error prone and without any guarantee of success.

An alternative solution for evaluating the power consumption profile is to estimate power instead of measuring it. Using this approach, designers may have a global idea whether the power consumed by a given design is compatible with the specified power budget. Clearly measurements provide actual values, while estimations do not. Nevertheless, if the fidelity and the accuracy of the chosen model is good enough to allow to opt for an optimal configuration, or at least to eliminate a set of possible configurations, then predictions can clearly improve designers' lives and shorten time to market.

Fidelity and accuracy concepts are strongly linked to the level of abstraction ones. To rapidly explore large design spaces, it is mandatory to raise the design abstraction level. Nevertheless, although very flexible, this approach may have severe drawbacks, especially in terms of power estimation. At system level, for example, a full system can be seen as a set of models without any implementation detail, and can be easily modified and simulated by simply adjusting specific parameters. Unfortunately, the accuracy in power estimation at that level is typically very low since hardware-dependent factors are not taken into consideration. Working at very low level of abstraction (at transistor or gate levels) provides very accurate models to the detriment of the simulation speed and flexibility [14]. At system level, to have accurate power prediction, flexibility still needs to be sacrificed. For instance, Paim et al. [28] presented a power-predictive environment specific for power-aware finite impulse response (FIR filter design) based on Remez algorithm. More flexible solutions focus on modeling, with the advantage of enabling power saving techniques. For instance, Nasirian et al. [25] presented an approach for power gating management in network-on-chip designs, adopting a cycle accurate simulator. In general, trade-off among complexity of the models and accuracy should be pursued. As an example, in the context of coarse-grained reconfigurable architectures, Fanni et al. [11, 29] were capable of reaching a good estimation accuracy in the proposed power models, while keeping the design exploration time low. This was performed by combining low level technological information and high level information related to the dataflow representation of the functionality implemented in the architecture.

In our work, we combine different abstraction levels in order to benefit from both a very fast simulation speed and from a good power estimation accuracy. We follow a component oriented approach. The design is considered as a set of interconnected components exchanging data, and each component is independently characterized

in terms of power, leveraging on two distinct models. Each model works on a set of features that is extracted from the component input binary data. The first model aims at determining power consumption from input features, whereas the second one maps the input data features to the output data features. After a building step, which consists of interconnecting all the components of the design, a system-level simulation is performed to propagate the data features throughout the system and to determine the contribution of each component in the total consumed power.

A major issue when dealing with modeling is to determine the correct level of abstraction and granularity of the components description that will guarantee a good trade-off between accuracy and exploration speed. A component may be an Intellectual Property (IP) block with complex functionality and structure, or it may also be a simple logic gate. Dealing with fine grain components, such as logic gates, provides designers with high accuracy and flexibility, but to the detriment of simulation speed and complexity. Working with coarse grain components, on the other side, enables faster simulation time at the price of poor accuracy.

In our work, to model the dynamic power consumption contribution in Application Specific Integrated Circuit (ASIC) designs, we work at the Register Transfer Level (RTL). We assume that functional specifications, for example, signal processing algorithms may be decomposed into a library of elementary components, such as adders, multipliers, registers, subtractors, Multiplier/Accumulator(MACs), etc. The methodology we are proposing is called NeuPow and leverages on Artificial Neural Networks (ANNs) to model power and signal propagation behavior of the considered components. NeuPow is an accurate, flexible and fast power estimation methodology that allows an early stage design space exploration of ASIC digital systems. NeuPow is currently capable of providing power estimation both at component and system-level. Components and their related ANN models constitute a library that can be used as a reference to explore various design configurations.

In a previous positioning work [27], we presented the basis of NeuPow that consists of characterizing a set of elementary components before evaluating the global power consumption of a full design. ANNs have been used to model power and behaviour of simple operators implemented in ASICs. A proof of concept of potential NeuPow benefits was carried out considering, as design under test, a complex multiplication. The Cadence®GPDK 45 nm technology library was used and no assessment on the methodology scalability for larger design was carried out. Building upon such preliminary study, in this work, we improve different elements of the methodology and provide a wider assessment. Below, the main contributions with respect to our previous positioning work [27] are summarized:

- the proposed models have been improved:
  - glitches have a significant impact on power consumption: they are now taken into consideration during the characterization phase to increase the final estimation accuracy;
  - also the clock frequency parameter has an impact on the estimations: we made an exploration of NeuPow behavior, while varying the frequency to include also this parameter in the power model;
- a wider exploration of the accuracy of the proposed methodology has been carried out:
  - a comparison among Neural Network based and regression based methods is presented;
  - a scalability study to evaluate the effect of activity propagation over cascaded layers in a circuit is discussed;
- the estimation time has also been evaluated and a noticeable acceleration with respect to the estimation procedures carried out with gate-level simulation is reported;
- assessment over a different design technology has been performed, leveraging on a 15nm ASIC technology.

The outline of the paper is as follows. Section 2 presents the context study and provides some generic background on the subject. In Section 3, related works are described. Section 4.1 describes the characterization methodology that has been proposed to build efficient power models. The modeling methodology is then described

in Section 4.2 and the global power estimation methodology is presented in Section 4.3. The proposed estimation approach has been tested on use cases that are described in Section 5.

## 2 CONTEXT STUDY

### 2.1 Power Consumption in Digital Systems

Optimization of power consumption is crucial for the design of low power digital devices. The total power consumption in digital circuits consists of static and dynamic power, as formulated in Equation 1. The static power consumption, given by Equation 2, is resulting from the leakage current $I_{lkg}$ (little non-ideal current that is always flowing within transistors) multiplied by the supply voltage $V_{DD}$. This term, when the circuit is powered on, is always present and independent from signals activity. The dynamic power, formulated as in Equation 3, depends on the switching capacitance $C_{load}$ (charge and discharge of the transistors during operation), the supply voltage $V_{DD}$, the design clock frequency $f$, and switching activity of the signals $SW$.

$$P_{Processing} = P_{Static} + P_{Dynamic} \tag{1}$$

$$P_{Static} = I_{lkg} \times V_{DD} \tag{2}$$

$$P_{Dynamic} = \frac{1}{2} C_{load} V_{DD}^2 \times f \times SW \tag{3}$$

Both $P_{Static}$ and $P_{Dynamic}$ depend on the technology, and more specifically, on the transistor size [20]. As transistors get smaller, $V_{DD}$ decreases and the operating frequency $f$ increases, due to general reduction of gate delay with size shrinking. This means that, on one hand, the dynamic power consumption decreases due to the reduced $V_{DD}$, but on the other hand, it has a positive trend due to the increase of $f$. By looking at Equation 2, we can notice that decreasing $V_{DD}$ implies a reduction of $P_{Static}$. However, a decrease in $V_{DD}$ requires reducing also the threshold voltage $V_{th}$ of the transistor to maintain a good performance. The reduction of $V_{th}$ has the effect of increasing the sub-threshold leakage exponentially, which in turns increases the static power. Therefore, with technology scaling, we generally assist to an increase of $P_{Static}$. Historically, the dynamic power being quadratically related to $V_{DD}$ as formulated in Equation 3, was considerably larger than $P_{Static}$ in terms of absolute values. Nevertheless, the effect of $V_{DD}$ decreased with the technology scaling, which led to a twist in this trend. It is very well known that while going from 90 nm technology to the next technology node, the contribution of the static power consumption starts to become no longer negligible (see Figure 1 in [21]). Nevertheless, we focus in our work on the dynamic power term. The main reason is that such term is highly system related. Optimal design choices and system composition have a direct influence on it. Considering the same resources with different topology choices, the static power does not change. On the contrary, the dynamic consumption may change, since it is dependent on the specific design and on the switching activity $SW$ dependence, as shown in Equation 3.

Key issue in designing low power digital devices is estimating the power consumption. However, this can be done at different levels of the design flow. A typical hardware design flow consists of the Register Transfer Level (RTL) modeling for a given design specification, followed by a design synthesis step, which maps the functional description into logic gates (netlist). Then, the design implementation step that maps the netlist to the hardware layout takes place. The most accurate estimation can be performed when the design is ready for tape-out, but the whole process requires long time. Thus, this option can be used as final verification of the power constrains, rather than for design choices. To be effective on the exploration of different design choices, power estimation should be performed at an earlier stage of the design flow. A first realistic estimation of the consumed power can be done in the post-synthesis phase, providing data activity information by means of post-synthesis timing simulation. The drawback of the post synthesis simulation is that the bigger is the design the longer is

the simulation time; furthermore, the synthesis itself may take long time. To explore $n$ different design choices, $n$ synthesis and $n$ simulations are required. Thus, an exploration of all the design possibilities and the related power performance would easily turn out in an extremely resource and time consuming process.

Being able to analyze the design at an early design stage would certainly save time. Commercial design suites have started to address this issue. For instance Cadence®has introduced the Joules RTL Power Solution [5], an RTL power analysis solution that offers the possibility to analyze the power consumption accurately during design exploration. However, this solution still requires to run 1) the design synthesis and 2) the power analysis every time the design is altered.

## 2.2 Neural Networks

Machine learning techniques are very efficient in pattern recognition problems, prediction, control, and classification. Those techniques are often invoked when characteristic features can appear anywhere in the input stream. One of them is based on Artificial Neural Networks (ANNs) that are supervised entities capable of learning from a set of examples.

These networks consist of algorithms that can be used to perform non-linear statistical modeling and provide an efficient alternative to logistic regression, the most commonly used approach for developing predictive models. These networks have a number of advantages, including requiring less formal statistical training, and the ability to establish complex and non-linear relationships between input variables. Moreover, these techniques have been massively used in the past and have demonstrated their efficiency.

ANNs are based on the connected neurons which represent the neurons in the biological brain. The connections among the neurons are like the synapses in the brain, and these connections are responsible to transmit the signals from one neuron to another. These networks consist of processing units called neurons that operate in parallel to solve complex computational problems. In this study, we consider basic Multi-Layer Perceptrons (MLP) feed forward networks that permit to model complex behaviors and may perform multi-dimensional functions approximation. Such networks have one or more hidden layers composed of neurons with non-linear transfer function and provide an output layer that implements output neurons with a linear activation function. Figure 1 shows a typical architecture of an MLP neural network.

In most of the neural models adopted in this paper, three layers have been used. Each layer receives its inputs from the precedent layer and forwards its outputs to the subsequent. In the forward phase, the hidden layer weight matrix is multiplied by the input vector $X = (x_1, x_2, x_3, \ldots, x_n)^T$ to compute the hidden layer output, as expressed in Equation 4.

$$y_{h,j} = f\left(\sum_{i=1}^{n} w_{h,ji} x_i - \theta\right) \tag{4}$$

where $w_{h,ji}$ is the weight connecting input $i$ to unit $j$ in the hidden neuron layer. $\theta$ is an offset termed bias that is also connected to each neuron. In order to train the networks, the well known back-propagation algorithm is often used. It consists of presenting a set of samples to the neural network, as well as a set of corresponding targets. During an iterative process, errors between the actual output and the desired target are back-propagated and all weights in the networks are modified according to their contribution to the error. The training process is generally stopped after cross-validation on another set of data denoted as validation set. This latter is primarily used to estimate the ability of an ANN to perform on unseen data. It aims at estimating how the model is expected to perform in general when used to make predictions on data that have not been used during the training process. After the training process denoted as the forward phase, the network shall be capable of estimating the correct output for any given input pattern.

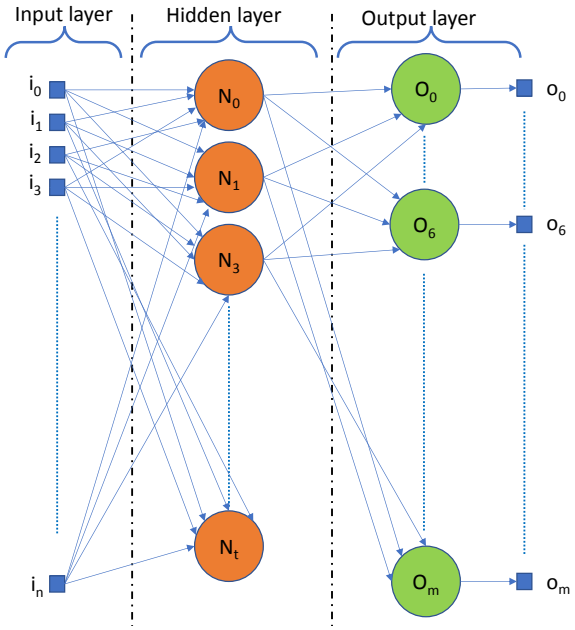


Fig. 1. Multi-Layer Perceptron (MLP) Architecture

## 3 RELATED WORKS

Several works dealt with the issue of modeling power consumption in digital circuits, with the goal of designing low-power circuits. Helms et al. [14] presented an efficient and accurate analytic per gate leakage model that takes all relevant leakage effects into account. It can predict leakage currents without needing any SPICE simulations for model application. Paim et al. [28] focused on a power-predictive environment for power-aware FIR filter design based on Remez algorithm, which enables fast and power-aware decision even in mathematical design level, reducing the power dissipation and the time-to-market of the chip. Fanni et al. [11, 29] studied an estimation model of power consumption to identify the regions of the design that can benefit from the application of power saving techniques. Nasirian et al. [25] modeled the behaviour of router buffers using queuing theory to evaluate the effect of power gating on the overall power saving and power penalty on network-on-chip. To evaluate the proposed algorithm, they adopted a cycle accurate simulator. However, generally speaking, all the mentioned works lack in terms of generality. In the example analyzed above, Paim et al. [28] focused on FIR filters, the work presented by Fanni et al. [11, 29] is valid for coarse-grain virtual reconfigurable systems, and the one from Paim et al. [28] for networks-on-chip.

Other researchers focused on more generic approaches. At the beginning, many authors exploited Look-Up Tables (LUTs) based approaches to model power [30], [12], [1], [13] and [10]. These techniques consist of tabulating values in order to estimate the global power of a design. To address the values that are stored in a table, different metrics may be used, such as the input transition density, input static probability, space-temporal correlations, etc. Generally, an additional method is also required to interpolate the missing values in the table. In this part, we review all the existing works that use the table-based power modeling technique in digital circuits. They demonstrated to be effective, reaching high estimation accuracy, but they also present some limitations as 1) they need a large amount of data to be stored, 2) the modeling effort is thus considered as moderate, and this factor depends on the number of attributes used in order to construct the table, 3) this technique is a memory

consuming method due to the large amount of the stored data, 4) the computational effort increases as the table grows, because of the dense search performed to get the proper power value for the given input entries. Thus, researcher started to focus on more effective approaches as Regression based techniques and Artificial Neural Networks (ANNs) based techniques.

Regression based techniques are used to map the power values (obtained from simulations or measurements) to specific parameters, such as the switching activity and static probability. Hence, this category focuses on bottom-up techniques based on a characterization phase generally performed at low-level of abstraction. Shang et al. [32] presented an adaptive regression method that is employed to reduce the problem of limited number of training sequences, by calling a gate-level simulator on a small number of cycles to improve the equation, achieving a relative error of 3.1%. Bogliolo et al. [3] proposed an advanced regression technique, based on regression trees, to improve the accuracy of the linear models. Control variables were defined to choose the most appropriate regression equations. An online characterization was also proposed to improve the power estimation accuracy of small combination circuits from 34.6% to 6.1%. Jevtic at al. [17] presented a power model for embedded DSP blocks of FPGAs, taking into account various signal statistics and multiplier sizes. The model was realized using a multi-variable regression over different power measurements and an accuracy of 7.9% has been achieved. Finally, several works focused on creating power models for basic operators, such as adders or multipliers [7, 9, 18] on FPGAs. While taking into account the switching activity, the operating frequency, the auto-correlation coefficient and the word length, a power model can be created as shown in [7], leading to an average error of 10% on FPGA. However, such approach does not consider the additional interconnections when estimating the power consumption of more complex design. In [9], the area and the power models for fixed point and floating point IPs were developed using linear regression. Here the area model is used to estimate the power by considering the number of slices, the memory blocks and the multiplier. Both the static and the dynamic powers can be estimated from this method, with an average error of 7%. Modeling operators is well-suited when estimating the power at a higher level of abstraction, leading to a fast exploration. However, such approaches do not consider the power consumption for a composite system composed of different components. Therefore, the exploration of the design space of the different possible configurations of a given design is not feasible.

Recent works that deal with the power modeling are based on machine learning approaches. ANNs are generally used to model power consumption. They can be adopted to model non-linear systems since they have the capability of learning the trends of input data automatically. In addition to this, they may achieve a very good accuracy with low complexity. Therefore, instead of using a mathematical model, neural models are suitable to be adopted in modeling problems for digital circuit power estimation. Borovyi et al. [4] exploited the neural networks for estimating the power consumption of digital processing systems. They used real hardware data from the ARM7TDMI processor to train the ANN. The limit of this approach is the lack of applicability, since results are valid for only this type of processor. Hsieh et al. [16] presented a power modeling technique of CMOS sequential circuits, based on the use of the recurrent neural networks (RNN). This work considers the non-linear characteristic of the power consumption distributions and the temporal correlation of the input data. The number of parameters necessary to estimate power consumption is about 8, and it requires a bit-accurate computation cycle at the input and at the output side to get power dissipation. Since power estimation requires the number of output transitions of a component, a behavioral simulation is needed at the beginning. Experimental results testify an error range of about 4.19%. This work is limited by two constraints: 1) the number of the necessary parameters that are needed to estimate the power consumption (which is about 8), 2) the application of the modeling technique on the CMOS circuits, without benefiting from the fact that these CMOS circuits may be composed of several fine grain hardware components that can offer more flexibility while exploring the power consumption of several coarse grain circuits. Hou et al. [15] exploited neural networks to estimate power of VLSI chips with respect to different specifications parameters such as: frequency, flash, ROM, RAM, GPIO, ADC, other integrated peripherals (DAC, Op Amp, Analog comparator, DMA, Hardware Multiplier, SVS), LCD segments,

timers, interface, package. However, they do not consider the data stimuli statistics that have significant impact on dynamic power consumption. Furthermore, they do not present any evaluation of the time that is required to perform the estimation. Compared to the precedent approaches, some works have chosen to consider a lower level of granularity by assuming that a full system may be seen as a set of interconnected components. For example, Lorandel et al. [23] focused on modeling power at IP level. In particular, they used ANNs to model both power consumption and signal activities of an IP block implemented in an FPGA. They demonstrated the effectiveness of their approach that estimate power consumption very fast (the minimum speed-up factor achieved by the neural models is about 11500×) and with a good accuracy (around 5%). Authors discuss the possibility of perform design space exploration of a global system, estimating the power consumption of an IP-cascaded system, but they did not went through this exploration.

After all, here are most of the limitations which are not addressed by previous works based on ANNs:

- not considering system-level power exploration (most of the works are applied on circuits and coarse grained blocks);
- not taking into account the glitches in the modeling phase, which have an impact on power consumption;
- missing an important factor like the clock frequency metric, which has an important impact on the dynamic power consumption;
- lacking the comparison between other existing modeling techniques.

To the best of our knowledge, our previous works were the first proposing a power estimation methodology that allows designers to explore various hardware solutions in terms of power consumption and performance [26, 27]. The proposed estimation methodology and power models, based on ANNs, were exploited to predict the power consumed by digital components implemented on FPGA or on ASIC platforms. In this paper we improve our previous positioning work [27] on ASIC by proposing more efficient models, by addressing all the aforementioned limitations.

## 4 METHODOLOGY

The aim of the proposed methodology is to provide a library of components, with associated power models that can be used for building signal processing applications. The advantage of this library of models is to have a flexible way to explore the design space in terms of power consumption. Indeed, the space of possible design solutions (different combinations of the components within the library) could be easily evaluated in terms of power consumption by means of the ANN models corresponding to the involved components. The library of components is built according to a fixed target ASIC technology and operating frequency (see Section 4.1). The corresponding library of models is built by exploiting the ANNs (Section 4.2). Finally, the components and their models are used to estimate the power consumption of composite systems (Section 4.3).

### 4.1 Characterization

The aim of the characterization phase of the proposed high level power estimation methodology is to specialize the adopted ANN models to correctly approximate the real behavior and the power of the considered components. The complete characterization flow is depicted in Figure 2. The flow starts with a common branch and then it is split into behavioral (left branch) and power (right branch) characterization. At first, the common branch requires describing the components involved in the system at the RTL level, by means of Hardware Description Languages (HDL), like Verilog or VHDL. Such RTL descriptions of the components are then synthesized according to the desired ASIC target technology and operating frequency. Synthesis is necessary to retrieve:

- the netlist, that is the description of the system and its components in terms of native cells, gates. The gate-level description is unique, in the sense that it is specific for the selected ASIC target technology.

- the Standard Delay Format (SDF) file, storing the timing data generated by the adopted EDA tool. Also the SDF is unique and design process-dependent, since it contains scaling, environmental, and technology parameters.

Post-synthesis timing descriptions, that are netlist and SDF file, are a good trade-off between speed and accuracy in power estimation for ASICs. They lie at the halfway between behavioural (not accurate, but fast) and post-implementation (more accurate, but slower) simulation and power estimation possibilities. The next step of the common characterization branch is the timing simulation that takes the netlist and the SDF file coming from the previous synthesis step as inputs. Moreover, it also receives the timing libraries of the chosen technology as input, which bring additional information about the timing of cells involved in the netlist, and a testbench to feed the netlist with input stimuli.

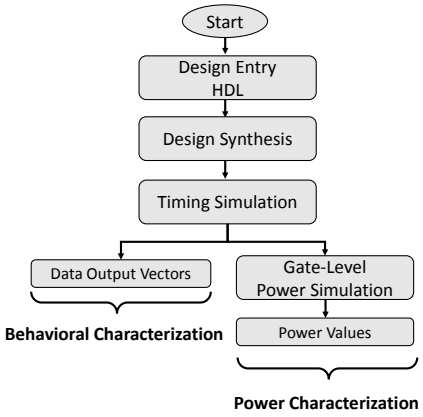


Fig. 2. Overview of the Characterization Methodology for the Models of Components.

During power characterization (right branch in Figure 2), power values corresponding to a certain input data stimuli (waveforms of the input signals) are extracted from the component under characterization. These values are gathered into two steps: firstly, a post-synthesis simulation is performed to calculate the propagation of output and internal signals due to the given input data stimuli. This step requires also, besides the input data stimuli, the netlist of the component and the related timing information. Secondly, the switching activity of the system, measured during the post-synthesis simulation, is taken as input of a power estimator tool that is provided by the Cadence®Genus Synthesis Solution [6]. The power estimator provides one scalar power value for each input data packet. The same power values are used to train, validate and test the ANN models used for power estimation. A detail of the power characterization branch is depicted in Figure 3.

The behavioral characterization (left branch in Figure 4) is intended to provide the data that are necessary to train, validate and test the ANN models in charge of mapping the outputs features according to the inputs' features. In particular, for each output, this phase evaluates the activity factor $AF$ and static probability $P_1$ features for a given input data stimuli, here referred also as packet. The $AF$ is related to the number of bit transitions during the simulation, while the $P_1$ to the probability of each bit to be set to one during simulation. Both $AF$ and $P_1$ are given per bit, so that the whole input/output features are vectors of per bit values. The conversion from waveforms (matrices of zeros and ones) to feature vectors is necessary on both input (for testing purposes) and output sides, as will be explained in details in Section 4.1.1. The behavioral characterization is performed by means of a post-synthesis simulation taking as input the input data stimuli, as well as the netlist of the component

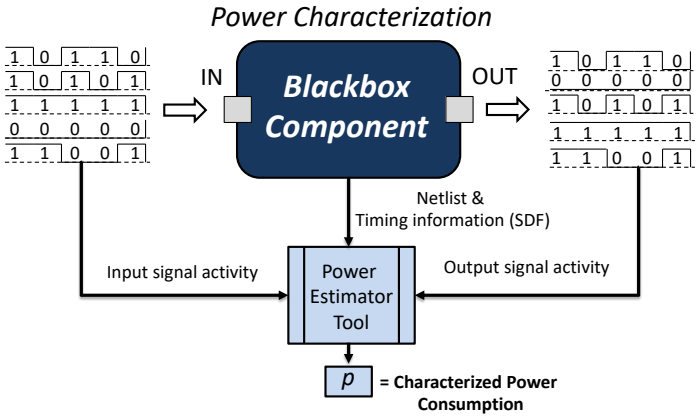


Fig. 3. Detail of the Power Characterization Branch

and the timing information about the involved cells and connections (SDF file plus timing libraries). Output signals are available at the end of the simulation as matrices of zeroes and ones (see Figure 4. The files containing information on values and timing of the input and output ports of the simulated component are then processed in order to extract *AF* and the *P* necessary for the characterization of the behavioral component models (more details are given in Section 4.2).

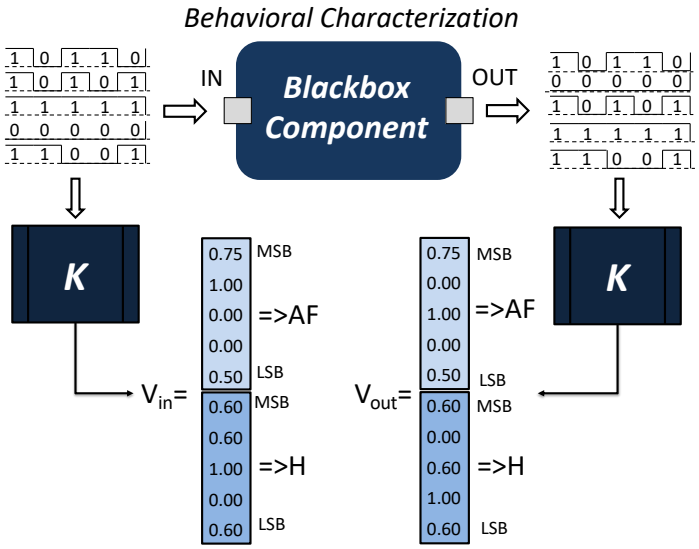


Fig. 4. Detail of the Behavioral Characterization Branch.

One of the key aspects of the proposed high level power estimation methodology is the adoption of ANNs for behavioral and power modeling of the system components. As deeply discussed in Section 2.2, ANNs usually

require a representative dataset to properly fit the desired behavior. Here a representative dataset, meaning a big and various enough set of data, implies a deep characterization of the components. Moreover, the dataset partitioning plays a role in ANNs performance: the splitting of a dataset into training, validation and test sets has an impact and thus, has to be defined carefully. Due to the importance of the dataset for the ANNs performance, in Section 4.1.1 the generation of the adopted input data for the components models characterization is discussed in details.

*4.1.1 Characterization Parameters.* To properly specialize the adopted ANN, it is of primary importance to provide a representative population of input data packets. An input data packet define a waveform that represents the value of each input digital signal in each unit of time throughout the considered time period. It can be also identified by two vectors of features, activity factor and static probability, that can be calculated according to the corresponding waveform of digital signals. One couple of values, one for activity factor and one for static probability, is expected for each bit of the considered signals. In order to provide a set of input packets that is representative enough, an automated script to generate data stimuli and provide the corresponding features has been developed.

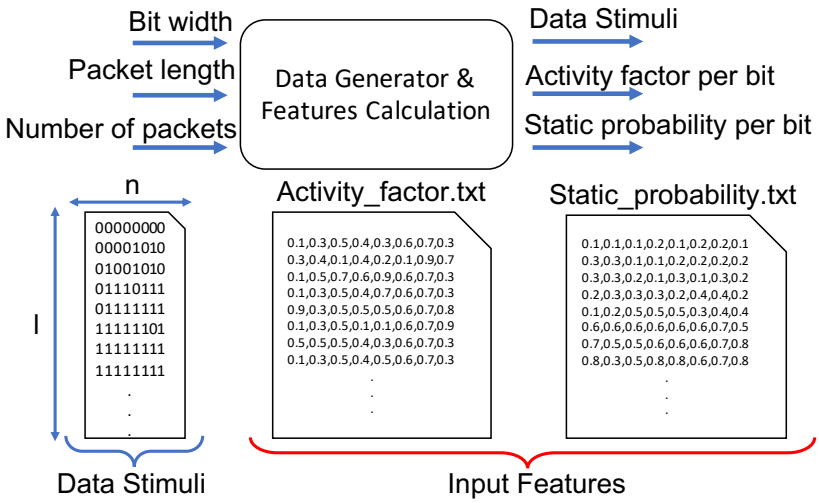


Fig. 5. Generation of the Input Dataset for Models Characterization.

As depicted in Figure 5, the script requires three different parameters:

- bit width $n$ that is the overall number of bits in input to the component under characterization. Dealing with an $m \times m$ bit adder with two $m$-bit inputs, then $n = 2 \times m$ bits;
- packet length $l$ that is the duration, in terms of clock periods for each packet;
- total number of packets $p$ to be generated.

The clock period is here defining the time unit considered for the observation period. The clock signal is not embedded within the model, but it is generated separately to synchronize the digital design. Thus, in its original implementation, each ANN model is bounded to a specific operating frequency. We will discuss, at the end of this work in Section 5.5, how the frequency affects the power estimation of the proposed methodology.

According to the developed script, each input packet is identified by a $2n \times l$ matrix of zeros and ones, representing the waveform of the component input signal, bit by bit (on the $n$ dimension), clock period by clock

period (on the $l$ dimension). Such matrix constitutes also one of the outputs of the script, the data stimuli that is the post-synthesis simulation input file (see Figure 5). The other two outputs of the script are the feature vectors corresponding to the data packet. These features, as already introduced, are the activity factor and static probability, both defined per bit. The activity factor, $AF[j]$ for the $j$-th bit can be expressed as in Equation 5:

$$AF[j] = \frac{\sum_{i=0}^{i=l-1}(x_i \oplus x_{i+1})}{l-1} \tag{5}$$

where $i$ and $j$ are column and row indexes of the data stimuli matrix respectively, and $x_i$ is the data (0 or 1) in the $j$-th row and $i$-th column. The static probability, $P_1[j]$ for the $j$-th bit can be expressed as in Equation 6:

$$P_1[j] = \frac{\sum_{i=0}^{i=l-1}(x_i)}{l} \tag{6}$$

Therefore, there are two factors ($AF[j], P_1[j]$) for each input bit of the characterized component and, in turn, there are $2 \times n$ features for each input data stimuli matrix. Note that passing from input data stimuli to input features, one dimension of the packets is somehow lost: the time. In fact, activity factor and static probability are both average values on the considered time period that is defined by the packet length $l$. Input features are not used during characterization phase since for both behavioral and power characterization input data stimuli is feeding the post-synthesis simulation phase. Anyway, they are used for training the ANN models since they constitute the inputs of both of them, while the reference outputs are provided by the characterization in terms of power values for the power ANN, and output feature vectors for the behavioral ANN.

Finally, the generated dataset is composed of $p$ input data stimuli matrices, each one corresponding to two vectors of features. The parameter $p$ is then responsible of the dataset size. The generation is performed randomly with the unique constraint of providing a uniform distribution of activity factor values.

*4.1.2 The glitch problem.* According to the definition of the activity factor per bit $AF[j]$ provided by Equation 5, it is also possible to define boundaries for such a metric. Being $l$ the packet length, that is number of clock periods per packet of data, $AF[j]$ can be better understood by comparing the corresponding waveform with the clock signal one. An activity factor $AF[j]$, that is equal to 1, means that the binary signal is flipping one time every clock cycle.

In the proposed methodology, the features are related to input and output bits of the modeled components. Therefore, the activity factor of the output bits depends on the internal behaviour of components. Ideally, considering synchronized signals at the input (data signals changing only on rising edges of the clock signal) and components with registered (flip-flops provided) inputs and outputs, output signals can flip once per clock period, at maximum. In this case, $AF[j] = 1$ constitutes an upper bound for the activity factor feature, as depicted by Equation 7:

$$0 \leq AF[j] \leq 1 \tag{7}$$

Static probability is also bounded since the percentage of time in which the signal can be high is related to the flips (value changes) of the same signal. In particular, when $AF[j] = 1$ then $P_1[j] = 0.5$ since the signal is always (each clock period) changing state during the observed interval, so the percentage of seeing it asserted is 50%. On the other hand, when $AF[j] = 0$ then $P_1[j] = 0$ or $P_1[j] = 1$ since the signal is not changing, but keeps its initial state that can be 0 or 1. The resulting boundaries for the static probability are then defined by Equation 8:

$$\frac{AF[j]}{2} \leq P_1[j] \leq 1 - \frac{AF[j]}{2} \tag{8}$$

In reality, even if we consider synchronized signals at the input side and components with registered inputs and outputs, the propagation of the signals from the input to the output of the component can introduce additional flippings, called glitches [13, 22]. This is even more evident if one or both assumptions, synchronized inputs and registered input/output, are not verified. In this case, the activity factor $AF[j]$ of the output bits can also reach the value 2, meaning that the output signal is flipping like the clock one, that is twice per clock period (from 0 to 1, and then back to 0). Then, considering glitchy signals, the boundaries for $AF[j]$ and $P\_1[j]$ are modified as shown by Equations 9 and 10:

$$1 < AF[j] \leq 2 \tag{9}$$

$$\frac{AF[j]}{4} \leq P_1[j] \leq 1 - \frac{AF[j]}{4} \tag{10}$$

Note that the boundary of 2 for $AF[j]$ strongly depends on how the cells are modeled in terms of timing and power behavior in the considered target technology. This value, for the 45nm technology, has been derived directly by the gate-level power estimator tool.

In order to properly generate input data packets according to the boundaries fixed by the glitchy or not glitchy nature of the generated signals, the generation script implements the algorithm, depicted in Figure 6. At first, an activity factor $AF[j]$ is randomly generated within the range $[0, 2]$. Then, if the generated $AF[j]$ is less than or equal to 1, the signal is classified as glitch free, and its boundaries are defined by Equation 8 to randomly generate the static probability $P_1[j]$. Otherwise, the signal is classified as glitchy and Equation 10 is considered during $P_1[j]$ generation.

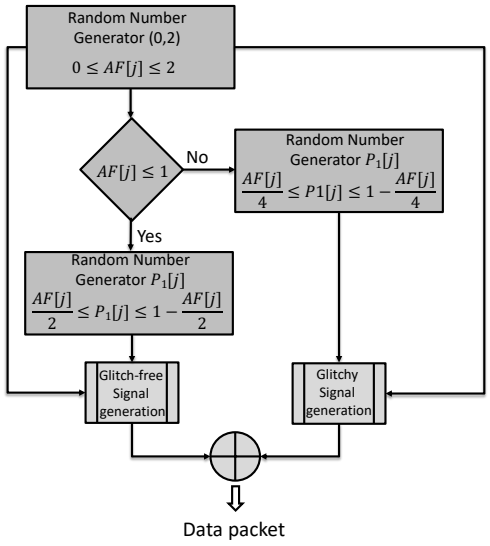


Fig. 6. Glitchy Input Data Stimuli Generation Algorithm.

Note here that the introduction of glitchy packets in the dataset adopted for models characterization (training set) has a direct impact on the accuracy of the models. Indeed, glitchy signals generated by a component are responsible for a significant amount of power consumption in the following components. We evaluated the difference between results obtained in our previous glitch free work [27] and the proposed glitchy methodology.

On the same complex multiplier test case adopted previously, the glitchy methodology is capable of achieving an average relative absolute error (%MRAE) of 2.6% instead of the 8.5% of the glitch free one obtained in Nasser et al. [27]. This means that, for the considered test case, by admitting glitchy signals in the ANN models characterization, it is possible to achieve power estimations that are more than 3 times accurate than considering only glitch free signals.

## 4.2 Modeling

As explained in Section 4.1, the proposed methodology is based on ANNs. The dataset coming from the behavioral and power characterization phase, discussed in Section 4.1, is used to train and test the networks. Since the dataset is partitioned into different subsets, only training datasets are used for specializing the ANN models in order to let them learn and fit with the desired functionality. In our case, two different ANN models have to be trained: the one accounted for behavior, that has to learn signals propagation mechanisms from the inputs to the outputs of the components, more specifically in terms of the corresponding features $V_{in}$ and $V_{out}$; and the one accounted for power, that has to learn the relationships between dynamic power consumption $P_{dynamic}$ and input data features $V_{in}$ of a given component. The functionality implemented by the behavioral and power ANN models can be represented as in Equation 11.

$$M : \begin{cases} f : V_{in} \rightarrow P_{dynamic} \\ g : V_{in} \rightarrow V_{out} \end{cases} \tag{11}$$

The addressed problem here is a multi-dimensional mapping problem, indeed the ANN models have multiple inputs and outputs. Furthermore, it is a problem that deals with continuous data since $AF[j]$ can assume any value between 0 and 2, and $P_1[j]$ can assume any value between 0 and 1. Being suitable for such kind of problem [31], a two layer feed forward ANN model with sigmoid neurons in the hidden layer and linear neurons in the output layer has been adopted. As can be seen from Figure 1, the adopted model has three layers:

- the input layer $i_k$, which its size is equal to the number of inputs of the model, thus twice the sum of bit width of the input signals of the corresponding component;
- the hidden layer $N_t$, involving all the sigmoid neurons whose number is usually set empirically;
- the output layer, composed of linear neurons, such that their number is fixed by the expected number of outputs of the specific model.

The functionality of the model is all concentrated within the hidden layer involving sigmoid neurons. Inputs of such neurons can be expressed as in Equation 12:

$$x_t = \sum_{k=0}^{k=n} (i_k \times w_{tk}) + b_t \tag{12}$$

where $w_{tk}$ is the weight of the $k$-th input for the $t$-th neuron and $b_t$ is the bias of the $t$-th neuron. Their outputs can be calculated through the sigmoidal non-linear activation function, as described by Equation 13:

$$N_t = \frac{1}{1 + \exp(-x_t)} \tag{13}$$

Similarly inputs of the output layer can be expressed by Equation 14:

$$z_j = \sum_{k=0}^{k=t} (N_k \times w_{jk}) + b_j \tag{14}$$

where $w_{jk}$ is again the weight of the $k$-th input for the $j$-th neuron and $b_j$ is the bias of the $j$-th neuron. Being linear neurons, the implemented function is a simple linear function in this case, so that the outputs of the output layer neurons can be obtained as in Equation 15:

$$O_j = z_j \tag{15}$$

Both the power and the behavioural ANN models have the same number of inputs and hidden neurons, as will be shown in Section 5. The main difference between the two ANN models is given by the output layer, and in particular, by the number of outputs. The power ANN, denoted as $f$ function in Equation 11, takes as input the vector of features $V_{in}$ corresponding to its input bits, while the output is a scalar value $P_{dynamic}$ expressing the dynamic power consumption that depends on the combination of inputs (see Figure 7).

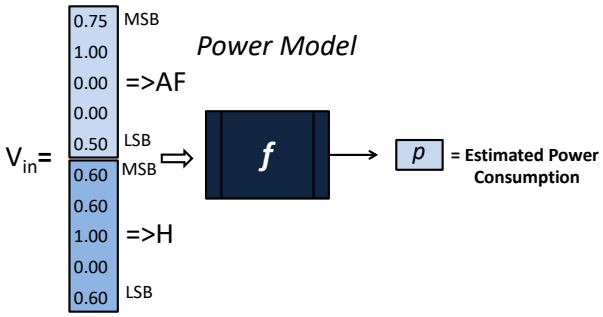


Fig. 7. Detail of Power Model Functionality.

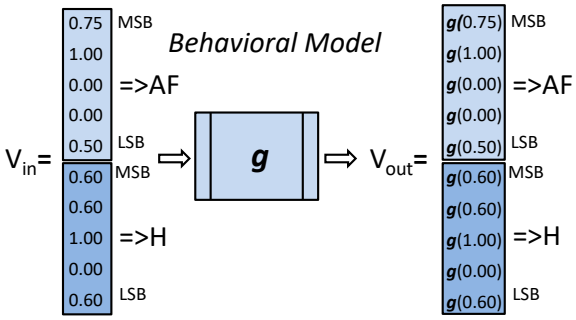


Fig. 8. Detail of Behavioral Model Functionality.

The output of the only neuron present in the output layer of the power ANN models can be then expressed as in Equation 16:

$$O_0 = P_{dynamic} \tag{16}$$

The ANN behavioral model $g$ is in charge of providing information about signals propagation from the inputs to the outputs of the component. As already said, the input of the behavioral model is the same of the power one: the vector of features $V_{in}$ related to input bits. While the output, in this case, is no more scalar being a vector of

features $V_{out}$ corresponding to the output bits, as shown in Figure 8. In this case, the outputs of the ANN can be defined as illustrated by Equation 17.

$$O_j = \begin{cases} AF[j] & if 0 \leq j \leq m/2 - 1; \\ P_1[j - m/2] & if m/2 \leq j \leq m - 1; \end{cases} \tag{17}$$

where $j$ goes from $0, ..., m$ and $m$ is doubling the number of bits of the component outputs. As discussed in Section 4.1, the features $AF[j]$ and $P_1[j - m/2]$, respectively, correspond to the activity factor and the static probability of the $j - th$ output bit.

### 4.3 Power Estimation

Our estimation methodology aims at offering to the designer the possibility of exploring the power consumption both at the component-level and the system-level. At the component-level, it is possible to use the developed power model $f$ in order to explore the average dynamic power consumption for the different possible input vectors, obtaining by this an average power. A first estimation of the average power consumption for a composite system, constructed as a cascade of components, as shown in the top part of Figure 9, can be expressed as in Equation 18:

$$P_{propagation-less}(system) = 1/n \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} f_j(V_{in}(i)) \tag{18}$$

where $i$ is the index of the input samples that ranges from 1 to $n$ and $j$ is the index of the component's model $f$ (power model) used to estimate the power consumption. This first estimation at the system-level is propagation-less, because it does not take into account that the input switching activity of a given component depends on the output switching activity of its predecessors.

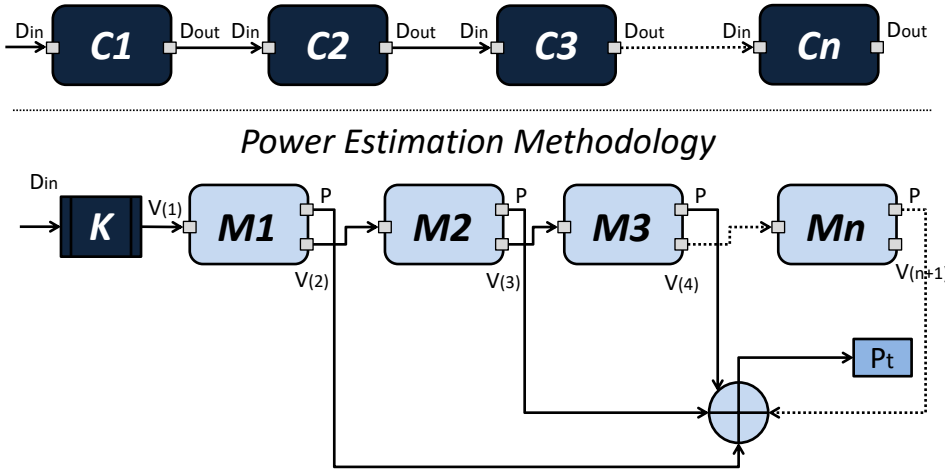


Fig. 9. Power Estimation Methodology

The proposed estimation method, called NeuPow, deals with the composite systems that involve several cascaded components or layers. The system topology can be described in terms of component's model $M$, as illustrated in Figure 9. The peculiarity of NeuPow is that, thanks to the behavioral model $g$, it propagates the

data stimuli $D_{in}$ among all the connected components or layers to achieve a more accurate power consumption estimation. This allows the designer to perform a faster and more accurate system-level power estimation, and to perform a more effective exploration of different design choices at an early design stage. The system-level power estimation $P_{NeuPow}$ corresponds to the accumulation of the dynamic power consumption for every individual component, as expressed as in Equation 19:

$$P_{NeuPow} = f_1(V(1)) + \sum_{j=2}^{j=n} f_j[g(V(j-1))]. \tag{19}$$

## 5 RESULTS ASSESSMENT

In this section, we demonstrate the results of the proposed estimation methodology both at the component and system-level. Starting from the accuracy, we also look at the scalability aspect and the frequency effect on the proposed modeling technology. We also discuss the advantages of our NeuPow methodology in terms of timing and its applicability on different technologies.

Estimation results are gathered after adopting a dataset generated by the stimuli generator script, composed of 10000 data packets ($p = 10000$ according to the parameters introduced in Section 4.1.1) that are divided randomly into: 80% for training, 10% for testing, and 10% for validation. For all the reported numbers, we adopted some of the available Cadence®tools: Genus Synthesis Solution for the RTL synthesis of the components, Incisive Enterprise Simulator for post-synthesis simulation required for the characterization phase, and again, Genus Synthesis Solution for power analysis purposes. Besides being used during characterization phase, the numbers resulting from these tools constitute also the term of comparison for most of the results, as will be shown in the sequel. Note that, unless explicitly specified in this section, the considered operating frequency for all the experiments is 100 MHz and the target ASIC technology library is the Cadence®GPDK 45 nm.

In Section 5.2, the results about ANN modeling are reported, showing how the ANN models have been dimensioned. Section 5.3 instead shows the assessment numbers of the proposed methodology at the component level, prior to the system level assessment presented in Section 5.4. Section 5.5 presents the frequency assessment and the model upgrade, so that the new models may take the operating frequencies into consideration for frequency-aware power estimation. Section 5.6 presents a time assessment of the proposed estimation methodology NeuPow. Finally, section 5.7 shows preliminary evaluation on a more recent technology, such as the 15nm FinFET-based Open Cell Library [24].

### 5.1 Use Cases Description

The basic library of components we have characterized and used in all the experiments is composed of the following elements: Multipliers (Mult), Adders (Add), Subtractors (Sub), Square, Multiply-accumulate (Mac), Register (Reg) and Shift and Clip (S&C). All the use cases that will be explored in the different sections below are built upon that library. In total, we have considered three use cases: 1) the complex multiplier, 2) the magnitude calculator, and 3) the image filter. Note here that the image filter use case is explored with four different design architectures. Hence, the use cases are shortly presented hereafter:

- Complex multiplier is used in several digital signal processing algorithms. Considering two complex numbers $Z_1$ and $Z_2$, it is possible to express them as $Z_1 = a + ib$ and $Z_2 = c + id$, where $a$ and $c$ represent the real part, while $b$ and $d$ are the imaginary part. The complex multiplication can be then written as $Z_1 \times Z_2 = (a \times c - b \times d) + (a \times d + b \times c)i$. It is possible to perform the complex multiplication using four $4 \times 4$ multipliers (Mult), one $8 \times 8$ subtractor (Sub) and one $8 \times 8$ adder (Add).
- Magnitude square calculator is used to compute the magnitude square of the resulting complex number after the complex multiplication, which can be expressed as $Real^2 + Imaginary^2$ (see Fig.10). Thus, it is

composed of the same components of the complex multiplier (4 Mult, 1 Sub and 2 Add) plus two components to calculate squared real and imaginary part (Square) and one final adder to compute the magnitude (Add).

- Image filter (see Fig.11) is a one-dimensional convolution filter adopted in different image and video processing applications. Basically, its composition depends on the number of taps (convolution kernel size): each filter stage is storing one value (Reg) and computing a multiplication between the input data and the corresponding kernel coefficient, and an accumulation, stage result plus the partial sum coming from the previous stage (Mac). In order to keep the result into meaningful values for an image (0-255), a final shift and clip (S&C) is required.

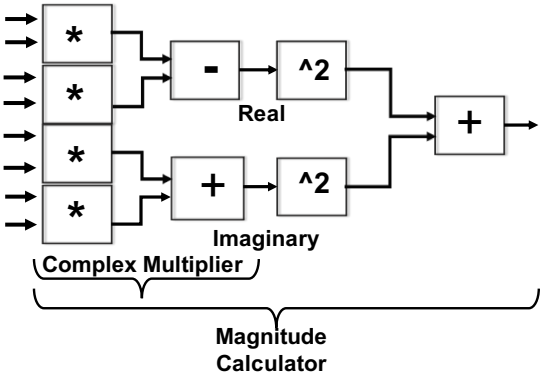


Fig. 10. Complex multiplier and magnitude square calculator

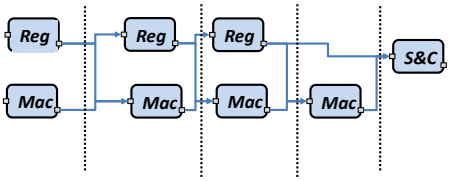


Fig. 11. Image filter (4layers)

## 5.2 Model Dimensioning

To properly dimension the ANN models adopted for power and behavioral estimation in the proposed methodology, we have performed an exploration on the unique parameter left free by the adopted ANN model (see Section 4.2): the number of neurons in the hidden layer. The metric used to evaluate the accuracy of ANN models is the Mean Square Error (*MSE*). This metric is the average squared difference between the estimated outputs and the desired ones (coming from Cadence®tools). *MSE* can be expressed as in Equation 20:

$$MSE = \frac{1}{N} \times \sum_{k=1}^{k=N} (T(k) - E(k))^2 \tag{20}$$

where $E$ is the value estimated using the proposed methodology, $T$ is the target value provided by Cadence®tools (Genus Synthesis Solution for power models and Incisive Enterprise Simulator for behavioral ones) and used

to train the model, and $N$ is the number of samples used for the test phase. Note that a small value of *MSE* is desired here: a zero *MSE* would mean that the model is able to estimate the considered output value with high accuracy regarding the examples that have been provided in its training phase, or at least as well as the adopted gate-level power estimation tool (Genus Synthesis Solution).

Table 1. Accuracy studies versus neural networks architectures.

| Models | ANN Architectures | MSE |
|---|---|---|
| $f$(Mac) | **72x25x1** | **1.2266** |
| | 72x50x1 | 1.4685 |
| | 72x25x25x1 | 2.2709 |
| $f$(Reg) | **18x25x1** | **1.4298E-4** |
| | 18x50x1 | 1.5435E-4 |
| $g$(Mac) | **72x25x36** | **6.3322E-4** |
| | 72x50x36 | 6.6931E-4 |
| | 72x100x36 | 6.8155E-4 |
| | 72x25x25x36 | 6.6944E-4 |
| | 72x50x50x36 | 6.5212E-4 |
| $g$(Reg) | **18x25x18** | **1.02141E-7** |
| | 18x50x18 | 1.02205E-7 |

Table 1 depicts how the *MSE* changes when the number of neurons in the hidden layer of the ANN model changes. For the sake of brevity, only Mac and Reg related results are reported. The number of hidden neurons selected to build the final ANN models of the components corresponds to the one with the smallest *MSE* value and is highlighted in bold for each component. For instance, considering the $f$ power model for Mac component, Table.1 demonstrates that changing the architecture of the neural networks in depth does not provide more accurate results, but less accurate ones due to the increase of the *MSE* from 1.2266 to 2.2709. To this end, 1 hidden layer with 25 neurons shows better results than 2 hidden layers (25$x$25). At the same time, considering the behavioral models $g$ shows also that the better configurations of the neural networks are in the case of a simple 1 hidden layer with 25 neurons. We think it is very difficult to theoretically prove that one hidden layer may outperform two hidden layers. It is also very difficult to determine a good network topology just by relying on the number of inputs and outputs. However, we believe that one hidden layer requires less data samples than a network with more hidden layers. Note that if the considered number of data samples is not enough, then the network will not have the possibility to generalize the model. In this regard, according to [19], an unnecessary increase in the number of hidden layers will lead to an over-fitting problem, and hence, a less network performance.

## 5.3 Component-Level Assessment

In this section, we discuss the result assessment of the proposed methodology achieved at the component-level. With this level, we intend that the results are related to the stand-alone components models, as if they are independent and isolated.

*5.3.1 Model Assessment.* To evaluate the accuracy of the adopted ANN models, besides the *MSE* introduced in Section 5.2 with Equation 20, we also use the regression index $R$, which measures the correlation between the estimated outputs (from the proposed ANN models) and the desired targets (from Cadence®Genus Synthesis

Solution for power, and from Cadence®Incisive Enterprise Simulator for behavior). $R$ can be expressed as in Equation 21:

$$R = \frac{N \sum TE - (\sum T \sum E)}{\sqrt{[N \sum T^2 - (\sum T)^2][N \sum E^2 - (\sum E)^2]}} \tag{21}$$

where $E$ and $T$ have the same meaning of Equation 20. In this case, having a value of $R$ close to 1 is desirable; this can imply a good correlation between the obtained $E$ and the desired $T$, and thus, the ANN model is more accurate.

The results for each component model are presented in the Table 2. As it can be noticed, they show good values for both considered metrics: the $MSE$ is generally small and $R$ is close to 1 for all the considered components. To prove the accuracy of the power ANN models, Table 2 reports also additional metrics as done in Bogliolo et al. [2]. The Relative-Root Mean Square Error ($R-RMSE$) is considered and can be expressed as:

$$R - RMSE = 100 \times \frac{\sqrt{MSE}}{P_{avg}} \tag{22}$$

where $P_{avg}$ is the average power of the test samples (intended as dynamic power). The amount of the power deviation of each power model ($RMSE$) can be defined as:

$$RMSE = \sqrt{MSE} \tag{23}$$

The results in Table 2 show that the ANNs are suitable to learn the relationship between the power dissipation and the input features. As for the power models, they can predict the power of the components with an error that is, in general, always less than ±1.4% (see $R-RMSE$ columns). The shift and clip (S&C) component represents an exception with an error of ±3.2%. This is due to the fact that the average power $P_{avg}$ of $S\&C$ is the lowest in the library, and hence, for approximately the same magnitude of $\sqrt{MSE}$, the corresponding $R-RMSE$ is larger.

Table 2. Modeling results for each component model (Values of $\sqrt{MSE}$ and $P_{avg}$ are in $\mu W$). Numbers before the component label indicate the number of bits per input signal of the same component.

| Models | MSE | R | $\pm\sqrt{MSE}$ | $P_{avg}$ | %R-RMSE |
|---|---|---|---|---|---|
| $f(4x4Mult)$ | 2.59E-2 | 0.99 | ±0.1609 | 11.89 | ±1.35 |
| $g(4x4Mult)$ | 1.3E-4 | 0.99 | - | - | - |
| $f(8x8Add)$ | 3.07E-2 | 0.99 | ±0.1752 | 15.20 | ±1.15 |
| $g(8x8Add)$ | 1.6E-4 | 0.99 | - | - | - |
| $f(16x16Add)$ | 0.1159 | 0.99 | ±0.3404 | 52.6360 | ±0.6467 |
| $f(8x8Sub)$ | 4.60E-2 | 0.99 | ±0.2145 | 16.35 | ±1.31 |
| $g(8x8Sub)$ | 1.8E-4 | 0.99 | - | - | - |
| $f(8Square)$ | 0.121365 | 0.99 | ±0.3484 | 47.6160 | ±0.7317 |
| $g(8Square)$ | 1.04166E-4 | 0.99 | - | - | - |
| $f(18x18Mac)$ | 1.2266 | 0.99 | ±1.1075 | 142.4904 | ±0.7772 |
| $g(18x18Mac)$ | 6.3322E-4 | 0.99 | - | - | - |
| $f(9Reg)$ | 1.4298E-4 | 0.99 | ±0.0120 | 24.4146 | ±0.0492 |
| $g(9Reg)$ | 1.02141E-7 | 1.00 | - | - | - |
| $f(18S\&C)$ | 5.87E-2 | 0.98 | ±0.2423 | 7.6077 | ±3.1849 |

*5.3.2 Comparison with other modeling Techniques.* In this section, in order to demonstrate the suitability of ANNs to address the considered power estimation problem, a comparison against a commonly used approach, the regression-based power modeling technique [7, 9, 17], is presented. This kind of approach approximates the power behavior of a component by means of a polynomial regression:

$$P = a.AC_{in} + b \tag{24}$$

where $P$ is the dynamic power of the component, $AC_{in}$ is the average (in terms of bits) switching activity at the input of a component, while $a$ and $b$ are parameters of the model that have to be fixed by means of a characterization phase, that is like the training for ANNs. To obtain such a regression-based method for a given component, it is then necessary to define $AC_{in}$ that, differently from ANNs activity factor, is not keeping per bit information. A component, composed of $n$ bits in input, can be characterized by an average switching activity expressed as in Equation 25:

$$AC_{in} = \frac{1}{n} \times \sum_{i=1}^{i=n} AF[i] \tag{25}$$

To build the regression-based power model, the power characterization of the different components has been carried out using RTL synthesis, post-synthesis simulations and power estimation offered by Cadence®tools. A database of the power consumption values for the different input data ($AF[i]$) has been built. The dataset size (10000), the operating frequency (100 MHz) and the target technology (Cadence®GPDK 45 nm technology) are the same for the ANN models.

Table 3 shows the results of the regression-based power modeling for the Mac, the Reg and the S&C, that are components of the image filter use case. Coefficients $a$ and $b$ are computed after the power characterization for each component, considering all the 10000 different input data.

Table 3. Regression-based Power Modeling Technique

| Models | Coefficients | | RMSE ($\mu W$) |
|---|---|---|---|
| | a | b | |
| $P_{MAC} = a \cdot AC_{in} + b$ | 1818 | 38.94 | 13.5121 |
| $P_{Reg} = a \cdot AC_{in} + b$ | 134.5 | 10.96 | 0.0130 |
| $P_{S\&C} = a \cdot AC_{in} + b$ | 25.32 | 5.073 | 1.2652 |

$R - RMSE$ represents a deviation relative to an average power consumption per component. Hence, it is worth to compute the percentage of the relative $RMSE$ to assess the accuracy of the modeling method, as in Equation 22.

Table 4 presents a comparison between the ANNs and the considered regression-based power modeling technique. The regression-based power modeling technique shows a very low accuracy when compared to ANN models. ANN models are able to guarantee an error that is always less than 3%, while the regression-based reaches up to 16% of error. This comparison demonstrates that ANNs are capable of overperforming commonly used power modeling approaches, like regression-based ones, for the considered use case and context.

## 5.4 System-Level Assessment

In this section, we discuss the system-level results of the proposed methodology. System-level means here that the results are related to sets of components connected together in order to compose more complex systems and functionalities.

Table 4. Regression-based Vs Artificial Neural Networks Power Modeling techniques

| Models | RMSE ($\mu W$) | | $P_{avg}(\mu W)$ | %R-RMSE | |
|---|---|---|---|---|---|
| | Regression-based | ANN-based | | Regression-based | ANN-based |
| MAC | 13.5121 | 1.1741 | 142.4904 | 9.4828 | 0.8240 |
| Reg | 0.0130 | 0.0122 | 24.4146 | 0.0532 | 0.0500 |
| S&C | 1.2652 | 0.2423 | 7.6077 | 16.6305 | 3.1849 |

*5.4.1 NeuPow Assessment.* This section is meant to assess the benefits of NeuPow, the proposed system-level power estimation methodology, using different case studies. In the result analysis below, we will compare power values obtained in three different ways:

- *Gate-Level Estimation*: power estimation provided by the Cadence®Genus Synthesis Solution according to the gate-level netlist of the system;
- *Propagation-Less Component-Based Estimation*: power estimation carried out with ANNs, as in Section 4.3 (see Equation 18). Component-level values of the average power consumption can be extracted using the same power model ($f$) adopted by NeuPow, and the total average power consumption of the system can be derived summing up the values of the power consumption associated to each component in the system without taking into account the behavioural models ($g$).
- *NeuPow Power Estimation*: power estimation carried out by NeuPow exploiting both power models per component $f$ and behavioural models per component $g$, as described in Section 4.3 (see Equation 19).

With respect to the *Propagation-Less Component-Based Estimation* qualitatively described above, we assume the following. The power model $f$ is used to estimate the power consumption with respect to an input vector $V_{in}$ for each component (see Equation 11). Consequently, the *Propagation-Less Component-Based Estimation* of a composite system can be expressed as in Equation 26:

$$P_{Component-based} = 1/n \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} f_j(V_{in}(i)) \tag{26}$$

where $i$ is the index of the input sample that ranges from 1 to $n$, and $j$ is the index of the $f$ component model used to estimate the power consumption.

Starting by the comparison between *Gate-Level Estimation* versus *Propagation-Less Component-Based Estimation*, results, presented in Table 5, show that the relative error of the *Propagation-Less Component-Based Estimation* is always below 22% for the considered cases. This error with respect to the target, the *Gate-Level Estimation*, is still high. The gathered values demonstrate that the usage of the *Propagation-Less Component-Based Estimation* has certainly limitations in terms of accuracy, but it could still be used to get the composite power consumption estimation of a system without using the behavioral model ($g$) described before, saving to the designer the burden of behavioral characterization and modeling for each component. Using *NeuPow Estimation*, as evidently shown in Table 5, the error with respect to the target never goes above 5.5%. In NeuPow, the behavioral $g$ model for each component makes it possible to propagate the switching activity among components that, in turn, leads to a more accurate power estimation on such components, and also for the global system.

*5.4.2 NeuPow Scalability Studies.* The accuracy of NeuPow has been analyzed also from a different perspective. This section is meant to assess what happens in terms of accuracy when the system depth and the number of cascaded layers change. For this, the components have been ordered in layers following a dataflow approach. A layer $N$ is then composed of components that receive values from the components in layer $N-1$ and that provide values to the components in layer $N+1$. In the following exploration, we considered different versions

Table 5. Comparison among Gate-Level, Propagation-less, and NeuPow Power Estimations

| Scalability Settings | Gate-Level (Cadence®Genus) ($\mu W$) | Component-Based (Propagation-less) ($\mu W$) | NeuPow ($\mu W$) | %RE (propagation-less) | %RE (NeuPow) |
|---|---|---|---|---|---|
| Complex Multiplier | 186.0959 | 154.9311 | 190.9553 | 16.7466 | 2.6112 |
| Magnitude calculator | 372.2309 | 302.7991 | 384.5835 | 18.6529 | 3.3185 |
| *Image Filter (4 Layers)* | 746.2769 | 650.8124 | 706.8775 | 12.7921 | 5.3619 |
| *Image Filter (6 Layers)* | 1135.000 | 984.6224 | 1142.900 | 22.0597 | 0.6167 |
| *Image Filter (8 Layers)* | 1526.600 | 1318.400 | 1576.600 | 13.6304 | 3.2765 |
| *Image Filter (10 Layers)* | 1916.000 | 1652.200 | 2011.200 | 13.7787 | 4.9582 |

of the image filter depicted in Figure 12 with different number of taps that correspond to different numbers of system layers: *4 taps/layers*, *6 taps/layers*, *8 taps/layers* and *10 taps/layers*. Figure 12, as an example, depicts the *4 taps/layers* image filter (top part of the figure) and a generic *N taps/layers* one to cover the cases from *6 taps/layers* to *10 taps/layers*. According to the image filter structure, it is possible to replace C1 by the Mac operator, C2 by the Reg, and C3 by S&C components.

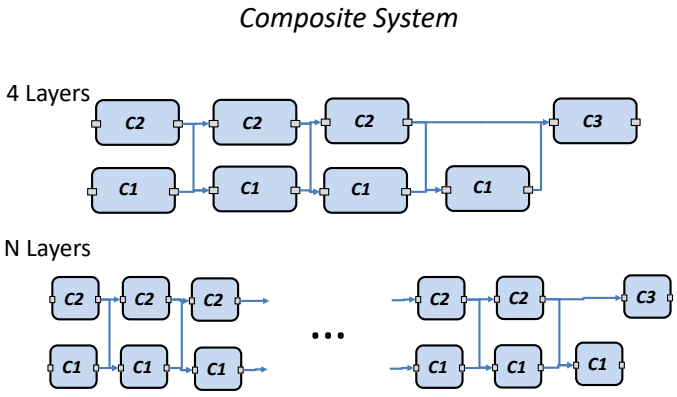


Fig. 12. Classical RTL description of a composite system

Following our modeling approach, it is possible to describe the image filter using the component models $M$ (see Equation 11) from the constructed library of models. To this end, Figure 13 presents the global system making use of component models. In this figure, $M1$, $M2$ and $M3$ respectively correspond to the models of a Mac, a Reg and a S&C.

Table 6 presents the scalability analysis, comparing *NeuPow Estimation* and *Gate-Level Estimation*. From this table, we can see that it is possible to appreciate how the maximum error of NeuPow is about 5.3%. Moreover, it is clear that this error is not growing with the deepness of the system, meaning that NeuPow performance is independent from the number of layers. In fact, the randomness of the error in sign and magnitude committed by the different ANN component models makes it possible to have even smaller errors for deeper systems in the considered use case with respect to the smallest (4 layers) design. In summary, this exploration shows that NeuPow provides a good estimation accuracy independently of the design size.

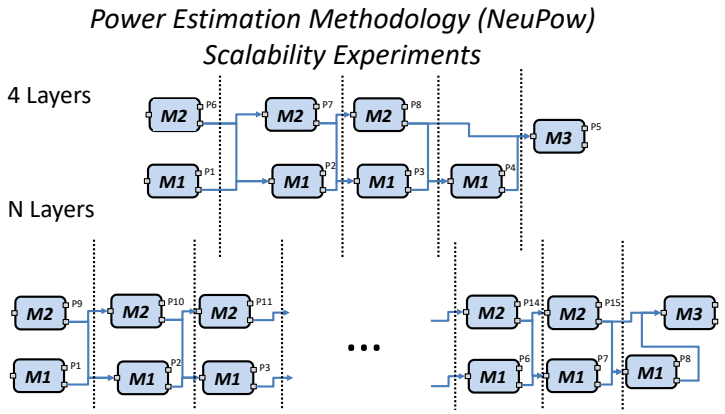


Fig. 13. NeuPow description of the digital filter based on the constructed library of models

Table 6. NeuPow: A Scalability Case Studies

| Scalability Settings | Gate-Level (Cadence®Genus) ($\mu W$) | NeuPow ($\mu W$) | %RE (NeuPow) |
|---|---|---|---|
| *Image Filter (4 Layers)* | 746.2769 | 706.8775 | 5.3619 |
| *Image Filter (6 Layers)* | 1135.000 | 1142.900 | 0.6167 |
| *Image Filter (8 Layers)* | 1526.600 | 1576.600 | 3.2765 |
| *Image Filter (10 Layers)* | 1916.000 | 2011.200 | 4.9582 |

## 5.5 Frequency Assessment and Model Upgrade

In digital circuits, the dynamic power consumption depends on several parameters, such as activity factor, static probability, operating frequency and technology. NeuPow was originally built while taking into consideration the first two metrics at a fixed operating frequency ($f_{clock} = 100MHz$) and target technology. In this section, we study how the dynamic power consumption changes with respect to the operating frequency. Generally speaking, the dynamic power and the frequency are proportional but, as depicted in Table 7, their relationship is not linear. For instance, the power consumption of a composite system is about $746\mu W$ at $100MHz$, while it is equal to $2547\mu W$ at $500MHz$. This means that the power is not scaled by 5 (not equal to $3730\mu W$), as it would be in case of linear relationship. If such linear approximation had been adopted for frequency aware power estimation, an error of 46.4% occurred. For this reason, we studied a more complex model that improves NeuPow, by taking into account the system operating frequency while predicting the power consumption. Figure14, illustrates such new model, called Frequency-aware Power Model.

Now to build the Frequency-aware Power Model, the baseline NeuPow is adopted, with all the steps that are related to the characterization, modeling and library construction phases, but with some modifications.This time the power characterization is performed at different frequency points, leading to different dynamic power consumption values. This frequency is added as a new input to the power model. Note here that the new component models are derived from the original $f$ and $g$ models. These models are characterized at $f_{clock} = 100MHz$. The output $V_{out}$ of the behavioral model is forwarded to the following component as before. The output of the power model $P_{100MHz}$ constitutes, instead, one of the input of a second ANN power model denoted by $k$, taking also

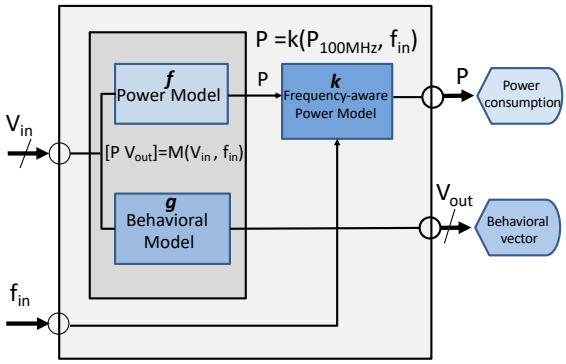


Fig. 14. Proposed Frequency-aware Power Model

as input the operating frequency $f_{in}$ (see Figure 14). The characterization of the new ANN power model $k$ is made using the power values coming by $f$ (trained with the same input features as used before to characterize the power) and by the frequency $f_{in}$, so that it can be possible to predict the new dynamic power consumption, taking into account the frequency points. The improved model $M$ is shown in Equation 27. Then, the new power model $k$ is also based on ANN, and it maps the dynamic power consumption $P_{100MHz}$ computed at $100MHz$ from the previous model $f$, and the input frequency $f_{in}$ to the corresponding, frequency-aware, dynamic power consumption.

$$M : \begin{cases} f : V_{in} \rightarrow P_{100MHz} \\ g : V_{in} \rightarrow V_{out} \\ k : P_{100MHz}, f_{in} \rightarrow P_{dynamic} \end{cases} \tag{27}$$

To evaluate the Frequency-aware Power Model, we consider as a test case the 4-layers image filter at different frequencies (100, 200, 300 and 500 $MHz$). Table 7 shows the achievable results using the improved NeuPowcompared with respect to the target Gate-Level power estimations. The error never exceeds 8.5%, but it is slightly increasing as frequency grows. By these preliminary data, it seems that, if the frequency is extremely different from the starting one (the one adopted for training $f$ and $g$ ANN models) the error could be high. A two-step model has been intentionally proposed in this work since our objective is to avoid a single complex network with heterogeneous inputs, specially that this may lead to an associated long learning processes and a loss of efficiency. We believe that the obtained errors are not due to the K frequency-aware power model, which usually provides a good accuracy while mapping the frequency dependency. However, the source of error goes back to the signals dynamic power consumed in the interconnections of a composite system. This can explain the overestimation of the gate-level estimation tools with respect to our method at high frequencies in all the studied cases. For this reason, Frequency-aware Power Model needs further investigations and refinement in the future.

## 5.6 Estimation Time Assessment

This section aims to compare the required time to run the power estimations using NeuPow with respect to that required using commercial tool-chains at gate-level as the Cadence®one. In the exploration below, we are using,

Table 7. Frequency-aware power estimations: Gate-level Vs NeuPow (upgraded).

| Frequency Configurations (MHz) | Gate-level (Cadence®Genus) ($\mu W$) | NeuPow (uW) | % RE |
|---|---|---|---|
| 100 | 746.2769 | 707.1985 | 5.2364 |
| 200 | 1331.400 | 1240.000 | 6.8650 |
| 300 | 1813.100 | 1663.000 | 8.2731 |
| 500 | 2547.800 | 2333.400 | 8.3994 |

for both *NeuPow Estimation* and *Gate-Level Estimation*, the same operating conditions (operating frequency, target technology and input data vectors) for each considered system configuration (that are the same of Section 5.4.2).

The power consumption for the different image filter design sizes (*4 layers*, *6 layers*, *8 layers* and *10 layers*) is extracted from both methods, and the time traces are obtained. For instance, each design configuration is simulated to get 10000 power values corresponding to the 10000 different input data vectors. Table 8 shows the results in terms of the estimation time (in clock counts *cc*) for each method. Hence, to explore the power consumption for all the four different designs that are considered, the gate-level power estimations require about $2.67e + 13cc$, whereas NeuPow needs $3.17e + 11cc$ only. This thus implies an acceleration factor of 84×. Note that the required time to characterize and to build the models in NeuPow is not accounted in the data reported in Table 8.

Table 8. Estimation time: Low-level gate level power simulations Vs NeuPoW power simulations

| Case studies | Gate-Level (Cadence®Genus) (clock counts) | NeuPow (clock counts) | Speed-up factor X |
|---|---|---|---|
| *4 Layers* | 4.20e+12 | 4.69e+10 | 89.55 |
| *6 Layers* | 5.75e+12 | 6.48e+10 | 88.73 |
| *8 Layers* | 7.44e+12 | 8.83e+10 | 84.25 |
| *10 Layers* | 9.35e+12 | 1.16e+11 | 80.60 |
| **Total clock counts cc** | **2.67e+13** | **3.17e+11** | **84.22** |

Despite the huge speed-up factor that NeuPow is capable of guaranteeing to its users when the library of components is available, the preparation of the library itself has a cost, and even the insertion of a new single component does not come for free. For any component addition, it is necessary:

- to characterize the component in terms of power and behavior, which implies performing several RTL synthesis, post-synthesis simulations and power estimations; and
- to train (specialize) the corresponding ANN models by feeding them with the provided characterization data.

Nevertheless, the advantage in the usage of NeuPow is still high. Indeed, if the designer manages to create a wide and heterogeneous component library, then he/she would always be able to get a substantial speed-up with respect to the classical approaches based on gate-level explorations. As an example of this work, with a simple library of 8 components, we were able to use NeuPow over 6 use cases.

### 5.7 Preliminary Assessment of NeuPow on a 15nm Technology

In this section, NeuPow is assessed on an additional target technology. Indeed, while all the previous results are referred to a 45nm technology, here follows an assessment on a different and more recent technology: the 15 nm FinFET-based Open Cell Library [24]. The differences between this technology and the previous one appear in the size of the technology point (15nm instead of 45nm) as well as in the kind of the gates used (FInFET instead of CMOS). The adoption of a different technology requires a complete execution of the whole characterization and modeling flow. Thus, we firstly performed model dimensioning (Section 5.2) whose results are omitted here for the sake of brevity. For the same reason, the analysis is limited to the image filter use case (discussed in Section 5), involving three different components: Mac, Reg and S&C. The component level results are shown in Table 9. The reported numbers show that the ANN models are still suitable for estimating the power of components synthesized on the 15 nm technology: maximum relative error is about 3.7%. As for the 45 nm technology, this occurs for the S&C component that has the lowest average power $P_{avg}$ among the library.

Table 9. Power Models on 15nm Technology.

| Models | MSE | $\pm\sqrt{MSE}$ (uW) | $P_{avg}$ (uW) | %R-RMSE |
|---|---|---|---|---|
| $f(Mac)$ | 0.370611 | ±0.6088 | 87.1459 | ±0.6986 |
| $f(Reg)$ | 0.0000219762 | ±0.0047 | 10.3939 | ±0.0451 |
| $f(S\&C)$ | 0.0203668 | ±0.1427 | 3.8859 | ±3.6726 |

Table 10 moves the analysis to the system level. In this table, a comparison among *Gate-Level Estimations* and *NeuPow Estimations* is reported for all the four design variants of the image filter use case: *4 taps/layers*, *6 taps/layers*, *8 taps/layers*, *10 taps/layers*. Considering this new technology, the relative error is always between 8% to 9.5% for all the filter configurations. This error has increased with respect to the 45 nm technology (the maximum relative error in such case was 5% for the *4 taps/layers* design), but is still below 10%. Note also that the error trend does not depend on the system layers, as it was for 45 nm technology case. This means that NeuPow scalability awareness is preserved.

Table 10. 15nm Technology Validation

| Test-Case | Gate-level (Cadence Genus) ($\mu W$) | NeuPow ($\mu W$) | %RE |
|---|---|---|---|
| Image Filter (4 Layers) | 424.5320 | 460.2543 | 8.4906 |
| Image Filter (6 Layers) | 664.1129 | 718.4044 | 8.1325 |
| Image Filter (8 Layers) | 896.7535 | 980.7820 | 9.3750 |
| Image Filter (10 Layers) | 1123.800 | 1225.600 | 9.0828 |

## 6 CONCLUSION

In this work, NeuPow, a system level ASIC power estimation methodology based on machine learning, is presented. The proposed methodology is capable of providing quick and accurate power estimations at a very early design phase. The approach is based on the adoption of Artificial Neural Networks (ANNs) to characterize the power and the behavior of a library of arithmetical components commonly used in signal processing applications.

It is possible to compose complex systems by connecting the components of the library and to estimate the corresponding power consumption. Such estimation leverages on behavioral ANN models to propagate the signal activity throughout components and on power ANN models that, taking as input signal activity coming from system inputs or from previous components, are capable of providing extremely precise power data.

Starting from a previous preliminary work, NeuPow has been firstly improved by including glitch effects within the ANN models characterization phase. Then, the proposed methodology has been deeply assessed under several aspects: 1) in terms of the adopted modeling technique, by comparing ANNs against widely adopted techniques in power modeling, such as regression-based ones; 2) in terms of accuracy, by considering several use cases; 3) in terms of scalability, by exploring different sizes of the design; 4) in terms of speed, by evaluating the time required to run the estimation of a large amount of design points. Moreover, some preliminary data on future directions of the work are given, especially with respect to the improvement of NeuPow by making it aware of the system operating frequency. Results on experiments made on different technology points have been also provided.

In the future, we intend to refine NeuPow further by evaluating more in details the performance of the proposed methodology when varying frequency and technology parameters. In particular, we intend to take the latter, or other parameters depending on it (e.g., operating voltage), as additional input, as we preliminary proposed in this work with frequency in order to provide frequency-aware technology-aware power estimation. In addition to this, deep learning approaches to enhance the modeling accuracy will be investigated.

## REFERENCES


[1] Massimiliano Barocci, Luca Benini, Allessandro Bogliolo, Bruno Riccó, and Giovanni De Micheli. 1999. Lookup table power macro-models for behavioral library components. In *Proceedings IEEE Alessandro Volta Memorial Workshop on Low-Power Design*. IEEE, 173–181.

[2] A. Bogliolo et al. 2000. Regression-based RTL power modeling. *ACM Trans. on Design Automation of Electronic Systems.* 5, 3 (2000), 337–372.

[3] Alessandro Bogliolo, Luca Benini, Giovanni De Micheli, Giovanni De Micheli, and Giovanni De Micheli. 2000. Regression-based RTL Power Modeling. *ACM Trans. Des. Autom. Electron. Syst.* 5, 3 (July 2000), 337–372. https://doi.org/10.1145/348019.348081

[4] Andrii Borovyi, Vasileios Konstantakos, Volodymyr Kochan, Volodymyr Turchenko, Anatoly Sachenko, and Theodore Laopoulos. 2008. Using neural network for the evaluation of power consumption of instructions execution. In *2008 IEEE Instrumentation and Measurement Technology Conference*. IEEE, 676–681.

[5] Cadence. 2019. Joules RTL Power Solution. https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html

[6] Cadence®. 2018. Genus Synthesis Solution. https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, 2018-08-28.

[7] Bedoui Mohamed Hedi Chalbi Najoua, Boubaker Mohamed. 2012. Accurate dynamic power model for FPGA basedimplementations. In *IJCSI International Journal of Computer Science Issues*, Vol. 9.

[8] Cisco. 2018. Internet of Things. https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf

[9] L. Deng, K. Sobti, and C. Chakrabarti. 2008. Accurate models for estimating area and power of FPGA implementations. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. 1417–1420. https://doi.org/10.1109/ICASSP.2008.4517885

[10] Yaseer A Durrani, T Riesgo, and F Machado. 2006. Statistical power estimation for register transfer level. In *Proceedings of the International Conference Mixed Design of Integrated Circuits and System, 2006. MIXDES 2006.* IEEE, 522–527.

[11] T. Fanni et al. 2016. Power and Clock Gating Modelling in Coarse Grained Reconfigurable Systems. In *Conf. on Computing Frontiers.*

[12] Subodh Gupta and Farid N Najm. 1997. Power macromodeling for high level power estimation. In *Proceedings of the 34th annual Design Automation Conference.* ACM, 365–370.

[13] Subodh Gupta and Farid N Najm. 2000. Power modeling for high-level power estimation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8, 1 (2000), 18–29.

[14] D. Helms, R. Eilers, M. Metzdorf, and W. Nebel. 2018. Leakage Models for High-Level Power Estimation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 8 (Aug 2018), 1627–1639. https://doi.org/10.1109/TCAD.2017.2760519

[15] Ligang Hou, Xinyi Wu, and Wuchen Wu. 2010. Neural network based power estimation on chip specification. In *The 3rd International Conference on Information Sciences and Interaction Sciences.* IEEE, 187–190.

[16] Wen-Tsan Hsieh, Chih-Chieh Shiue, and C-NJ Liu. 2005. A novel approach for high-level power modeling of sequential circuits using recurrent neural networks. In *2005 IEEE International Symposium on Circuits and Systems*. IEEE, 3591–3594.

[17] R. Jevtic and C. Carreras. 2010. Power Estimation of Embedded Multiplier Blocks in FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18, 5 (May 2010), 835–839. https://doi.org/10.1109/TVLSI.2009.2015326

[18] Tianyi Jiang, Xiaoyong Tang, and Prith Banerjee. 2004. Macro-models for High Level Area and Power Estimation on FPGAs. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI '04)*. ACM, New York, NY, USA, 162–165. https://doi.org/10.1145/988952.988992

[19] Saurabh Karsoliya. 2012. Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *International Journal of Engineering Trends and Technology* 3, 6 (2012), 714–717.

[20] Michael Keating, David Flynn, Rob Aitken, Alan Gibbons, and Kaijian Shi. 2007. *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated.

[21] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. 2003. Leakage current: Moore's law meets static power. *Computer* 36, 12 (Dec 2003), 68–75. https://doi.org/10.1109/MC.2003.1250885

[22] X Liu and M. C. Papaefthymiou. 2002. Incorporation of input glitches into power macromodeling. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, Vol. 4. IV–IV. https://doi.org/10.1109/ISCAS.2002.1010590

[23] Jordane Lorandel, Jean-Christophe Prévotet, and Maryline Hélard. 2016. Efficient modelling of FPGA-based IP blocks using neural networks. In *2016 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 571–575.

[24] Inc.®Nangate. 2019. NanGate FreePDK15 Open Cell Library. https://http://www.nangate.com/?page_id=2328, 2018-08-28.

[25] N. Nasirian, R. Soosahabi, and M. A. Bayoumi. 2018. Probabilistic Analysis of Power-Gating in Network-On-Chip Routers. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2018), 1–1. https://doi.org/10.1109/TCSII.2018.2863268

[26] Yehya Nasser, Jean-Christophe Prévotet, and Maryline Hélard. 2018. Power Modeling on FPGA: A Neural Model for RT-level Power Estimation. In *Proceedings of the 15th ACM International Conference on Computing Frontiers (CF '18)*. ACM, New York, NY, USA, 309–313. https://doi.org/10.1145/3203217.3204462

[27] Yehya Nasser, Carlo Sau, Jean-Christophe Prévotet, Tiziana Fanni, Francesca Palumbo, Maryline Hélard, and Luigi Raffo. 2019. NeuPow: Artificial Neural Networks for Power and Behavioral Modeling of Arithmetic Components in 45Nm ASICs Technology. In *Proceedings of the 16th ACM International Conference on Computing Frontiers (CF '19)*. ACM, New York, NY, USA, 183–189. https://doi.org/10.1145/3310273.3322820

[28] G. Paim, L. M. G. Rocha, T. G. Alves, R. S. Ferreira, E. A. C. da Costa, and S. Bampi. 2017. A power-predictive environment for fast and power-aware ASIC-based FIR filter design. In *2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*. 168–173.

[29] F. Palumbo et al. 2016. Modelling and Automated Implementation of Optimal Power Saving Strategies in Coarse-Grained Reconfigurable Architectures. *JECE* 2016 (Nov. 2016), 5.

[30] A. Raghunathan, S. Dey, and N. K. Jha. 1996. Register-transfer level estimation techniques for switching activity and power consumption. In *Proceedings of International Conference on Computer Aided Design*. 158–165. https://doi.org/10.1109/ICCAD.1996.569539

[31] F. Scarselli and A. C. Tsoi. 1998. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural networks* 11, 1 (1998), 15–37.

[32] Li Shang and N. K. Jha. 2001. High-level power modeling of CPLDs and FPGAs. In *Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001.* 46–51. https://doi.org/10.1109/ICCD.2001.955002