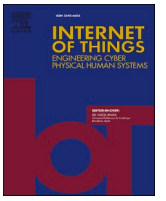
Internet of Things 21 (2023) 100655

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



Review article

Processor power and energy consumption estimation techniques in IoT applications: A review

P.Y. Dibal [a,*], E.N. Onwuka [b], S. Zubair [b], E.I. Nwankwo [b], S.A. Okoh [b], B. A Salihu [b], H.B. Mustaphab [b]

[a] Computer Engineering Department, University of Maiduguri, Nigeria
[b] Telecommunication Engineering Department, Federal University of Technology (FUT) Minna, Nigeria

ARTICLE INFO

Keywords:
Energy
Internet-of-Things
Power
Processor

ABSTRACT

The energy efficiency of IoT nodes remains the dominant factor for effective IoT solutions that will meet the challenges of the 21st century, especially in the drive towards a carbon-neutral world through net-zero targets. Microprocessors/microcontrollers are devices that perform entire operations of IoT devices. Therefore, the power and energy consumption of these processors directly reflects the power consumed by the IoT devices they drive. An accurate estimation of the power and energy consumption of the processors is vital for the development of energy-efficient IoT solutions because IoT devices are designed to operate in remote locations for long periods without human intervention. It is against this backdrop that this paper which is expected to serve as a guide for researches and IoT node/application developers in selecting the best technique for an IoT use-case, presents a review of processor power and energy consumption estimation techniques starting from the lowest level of abstraction to the highest level of abstraction. The review involves a detailed discussion of estimation technique methodologies for an abstraction level, and where applicable, generalized methodologies which cover the most approach used for an abstraction level are covered. The existence of overlaps and the impact of processor duty cycles on the techniques were discussed. A comparison of the strengths and weaknesses of each technique was made, from where register-transfer level and instruction level techniques are shown to be resilient against errors that occur from poor input signal conditioning. Future directions for the development of estimation techniques are also presented as recommendation.

1. Introduction

The Internet-of-Things (IoT) is a network computing paradigm that connects electronic devices distributed across different geographic locations for monitoring, data acquisition, and remote assessment of systems and environmental parameters [1–4]. IoT is envisaged to be a key enabling technology in the drive toward Net-zero infrastructure, smart cities, and climate-smart agriculture (CSA). To work effectively as an enabling technology, IoT must be energy-efficient. Thus, huge responsibility lies on the IoT node and application developers in ensuring that their product meets the required level of energy efficiency. To realize this, the efficiency of the IoT system being developed must be estimated using an appropriate estimation technique during critical milestones in the

* Corresponding author.
E-mail address: yoksa77@gmail.com (P.Y. Dibal).

https://doi.org/10.1016/j.iot.2022.100655

Available online 9 December 2022

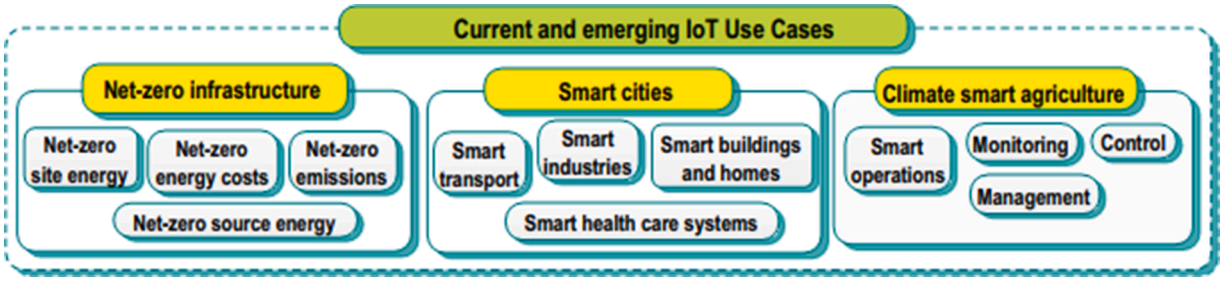


Fig. 1. Three generalized use-cases of IoT.

development process. It is against this backdrop that this paper presents a systematic review of power and energy estimation techniques in processors at different levels of abstractions to make available to the readers, the methodology, complexity, and likely challenges in using an estimation technique. An extensive review of literature shows that a one-stop review paper that discusses all the levels of abstraction in processor power and energy consumption estimation as presented in detail in this paper is currently lacking. Therefore, this review paper aims to fill the gap by equipping the reader or the developer with the necessary information about estimation techniques that will be required in making the right choice in the selection of a processor that plays a dominant role in IoT nodes and applications; these nodes and applications are envisaged to be the drivers of a sustainable future as shown in Fig. 1 [5,6]. The link between IoT and generalized use-cases in Fig. 1 covers most of the conceivable areas of current and future IoT applications. Net-zero infrastructure as shown is an emerging use-case for IoT, while smart cities and climate smart agriculture (CSA) are current use-cases for this technology.

The increase in the standard of living experienced in the last decade has led to a rapid increase in the quality and number of infrastructure projects in different regions of the globe [7,8]. This development has had an impact on the environment, especially in the context of carbon footprint, which has been directly linked to the dual problem of climate change and global warming. However, these problems can be kept under control by a transition to net-zero, where the rate of emission of greenhouse gases to the atmosphere is equal to the rate of removal of such gases. A candidate solution in literature is IoT, which is showing promising results toward net-zero infrastructure, and it is envisaged to play a leading role in that regard. Net-zero emissions are the major challenge in net-zero infrastructure, and the path to attaining it is quite narrow as it requires the deployment of efficient energy technologies controlled by intelligent IoT devices if the target is to be met [9,10]. To further corroborate this position, the energy trends forecast for net-zero emissions scenario by [9,11] puts the decline in coal use at 90% in 2050 from the levels in 2020; 30% cut in methane emissions by 2030; between 2020 and 2050, oil demands will fall by 75%; a drop of 55% is expected to occur for natural gas between 2020 and 2050. These projections, which form an active decarbonization process, can be achieved through IoT-driven energy-efficient systems [12]. Net-zero energy costs, along with site energy and source energy represent the drive towards significantly reducing the direct and indirect energy consumption of industries, and organizations [13].

Smart cities are a current use case for IoT applications, and they can be defined as complex sociotechnical structures whose existence is the result of the convergence of human actors and digital devices [14]. The human actors include citizens and city operators. The digital devices include city sensors and actuators that service transportation, industries, organizations, homes, and healthcare systems. IoT applications have a central role to play in the optimal operation of the digital devices for a city to be smart, and the degree of application of IoT in the operation of the digital devices determines the degree of the sustainability of a city. In smart transport, IoT applications enable the optimal management of traffic, route decision, and resource allocation while at the same time providing traveler information to commuters in an efficient manner [15–17].

The application of IoT in smart industries includes the real-time monitoring of production processes, and automatic adaptation to changing materials conditions and customer's needs [18–20]. Equipped with an appropriate AI algorithm, IoT can make accurate predictions when a piece of industrial equipment will fail; this will make it possible to perform resource-efficient preventive maintenance [21,22]. In smart homes, IoT is the nexus between energy conservation and optimal appliance management. Through specialized AI algorithms, IoT applications can monitor and regulate heating, cooling, and lighting systems in homes [23,24]. Such an approach significantly scales down the impact a home has on its energy mixes like supply grid and renewables. The application of IoT in healthcare systems has shown rapid response to patients' needs by doctors through wearable devices which serve as IoT nodes. With these devices, patient records can be accessed, vitals can be remotely monitored and assessed remotely in real-time, and timely interventions can be made on-demand. This paradigm has revolutionized healthcare delivery [25–28].

Climate Smart Agriculture (CSA) is a use-case in which IoT is making deep penetration, and it is defined as the integration of traditional agricultural practices with data-driven and data-acquisition technologies like IoT for the sole aim of developing an efficient farming practice that is resilient to climate change, guarantees food security, and having minimum impact on environmental resources [29–31]. Smart operations are the aggregation of farming activities in which IoT plays a central role such that the operations are performed optimally and efficiently [31,32]. The operations involve a data-driven synchronization of complex farming activities in a way that maximum productivity can be achieved with the fewest farming tools and other inputs. In CSA, monitoring is a data-intensive task in which readings of a variety of parameters (temperature, humidity, soil pH, fertilizer concentration, etc.) central to a good output are made periodically by IoT sensor nodes. The readings are evaluated using AI algorithms to determine the appropriate response to be made [33–35]. Control in CSA involves the use of IoT to efficiently regulate the number of user inputs like water and

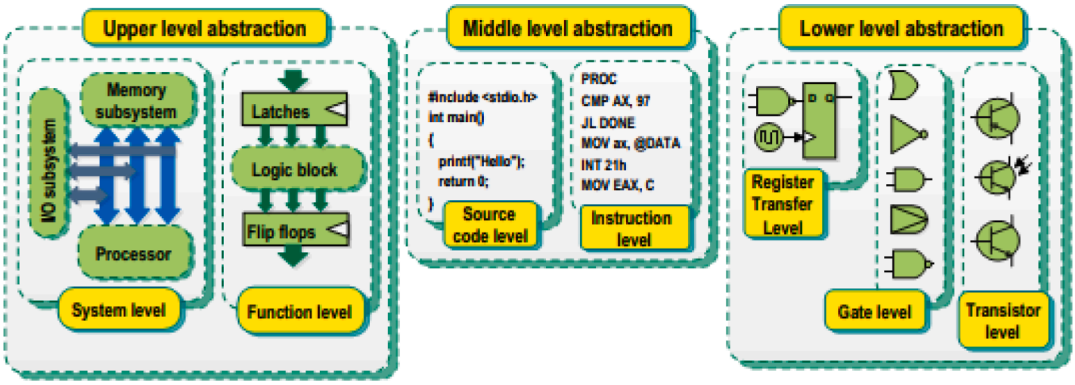


Fig. 2. A taxonomy of power and energy consumption estimation techniques for processors.

fertilizer so that farm yields which are within acceptable standards can be derived [36,37]. Control also implies the use of IoT to optimally apply insecticides and herbicides to create conditions that ensure that plant diseases, pests, and weeds are unable to thrive [36]. One of the major issues which CSA addresses are the wastage of scarce resources like fresh water in irrigation. By using IoT, efficient management of farms is achievable because IoT powers data-driven decisions which determine the best and optimal application of water, fertilizer, pesticides, and other inputs which produce high yields [38–40]. This is very important due to the rising cost of fertilizer, and the increasing pressure on available freshwater arising from population explosion.

Having looked at the use cases for IoT, an avid reader would like to see the objective of this paper in the context of processor power and energy consumption estimation, and IoT applications. The answer lies in identifying the benefits of estimating the power and energy of the processors. These benefits are itemized as follows:

(i) Every processor power and energy consumption estimation technique has its strength and weakness. It is therefore vital to have a sound knowledge of each estimation technique with a view of how they can, on a use-case basis, be selected for the accurate estimation of the power and energy efficiency of IoT applications.
(ii) The estimation of processor power and energy consumption through an appropriate technique creates the possibility of selecting suitable processors for the development of IoT products and applications which have good power and energy characteristics; this will guarantee the industrial viability of such applications and their environmental profitability in the context of carbon footprint reduction.
(iii) Advancements in digital electronics have significantly increased the functionality of processors which seamlessly execute the expanding functions of IoT applications. With this reality comes the possibility of many ways by which IoT applications can perform certain functions. However, not all routes taken in developing IoT applications will yield the same level of processor energy efficiency; it is in this context that an appropriate power and energy consumption estimation technique can be used in estimating the numerical value of the energy efficiency of an IoT application. Armed with this information, an appropriate IoT application development route that yields the best power and energy efficiency can be selected.
(iv) An energy-efficient IoT application developed using accurate power and energy consumption values obtained from an appropriate estimation technique translates into the development of IoT nodes that have a prolonged lifespan while operating in remote locations.

Consequently, it is appropriate to develop a taxonomy of the known techniques that are used in the estimation of processor power and energy consumption. Such a taxonomy is shown in Fig. 2 where techniques vary in abstraction levels i.e. from upper level abstractions to lower level abstractions.

From the objective of the paper and the foregoing discussions on the use-cases of IoT, it is apparent that the quest to achieve high standards of living within the context of a sustainable future will rely heavily on IoT. Through a fusion with machine learning, IoT is placing within our reach, the creation of a carbon-free world whose resources are sustainably utilized for the advancement of human society. Therefore, IoT is tangential and important to a sustainable future, and it is against this backdrop that this paper will focus on one of the key enablers of IoT technology i.e. power and energy consumption. The reason for this is that most IoT nodes operate remotely over an extended period; it is therefore expected that such nodes will be able to utilize their power for as long as possible without human intervention. To set the tone for the review, the paper is organized as follows: Section 2 reviews the role of power and energy in ensuring the viability of IoT technology and solutions. In Section 3, the factors responsible for the power consumption in processors are presented; this section lays the basis on which the power estimation techniques are discussed. Section 4 presents a taxonomy and review of power and energy consumption estimation techniques starting from the lowest level of abstraction to the highest level of abstraction. Section 5 discusses overlapping factors and effect of power duty cycles in power and energy consumption estimation. Section 6 highlights the advantages and disadvantages of each of the estimation techniques and also highlights future research directions. While Section 7 concludes the paper.

Table 1
Differences between power and energy.

Power	Energy
It is the rate at which the transfer of energy takes place	It is the cumulative amount of work performed over a given period
It is measured in Watt	It is measured in Joule
Cannot be converted from one form to another	Can be converted from one form to another
Cannot be stored	Can be stored
It is an instantaneous quantity	It is a time quantity

2. Power and energy as enablers of IoT technology

Power and energy consumption is arguably one of the most important performance metrics an IoT node can possess, and its efficient utilization implies that the IoT can work remotely over a long period. An energy-efficient IoT node is a node that can perform a set of tasks with minimal energy. In dealing with power and energy consumption, it is important to differentiate between power and energy, especially in the context of work. While power can be described as the rate at which work is performed or the rate at which energy is consumed; energy on the other hand is the total amount of work performed over a given period. These definitions imply that the reduction in CPU power consumption through reducing CPU performance does not translate into energy savings/efficiency because a longer time will still be required to complete the execution of a given set of tasks using the same or even more amount of energy [41, 42]. From the foregoing definition, the power and energy consumption of an IoT device determines its viability, and it is against this backdrop that the ability of an IoT device to conserve energy has received much attention from the research community. To further distinguish power from energy, Table 1 shows some key differences between power and energy.

Several factors are responsible for power consumption in an IoT node, and they include data communication between an IoT node to other nodes or a centralized base station, the type of microprocessor used in the IoT node, associated peripherals, and sensing operations. Several attempts have been made in literature toward conserving the energy of an IoT node, and they include energy harvesting protocols [43,44], application of optimization in routing protocols and communication link status, optimal sleep & wake up strategies based on network traffic, and data reduction based on optimization of network topology [45–48]. While these approaches have received a lot of attention in literature, a salient factor that is equally important but has not received equal attention is the power and energy consumption of the processors which run and control all the operations of the IoT device [49].

Energy conservation in IoT node from a processor perspective is based on the known fact that the CPU workload of the processor determines the energy consumption of the IoT node per time, and the CPU workload is determined by the number of CPU cycles necessary for the execution of an application [50]. Hence, energy can be conserved by an IoT node from a processor perspective through the development of applications that have high throughput. This will ensure that fewer CPU cycles will be required for the execution of applications; this will result in a lower workload and better energy savings. This information is critical because there are different processors from different manufacturers with varying performance; some processors perform better than others in running certain types of applications under certain ambient conditions. As a result of this, an IoT development and deployment process must be able to use an appropriate power and energy estimation technique to determine the suitability of an IoT node for a task in view based on its central processor. It is against this backdrop that the following sections will present a systematic examination of processor power and energy estimation techniques starting with factors that affect power consumption in processors.

3. Power consumption factors in processors

There are two broad categories of factors that affect the power consumption of processors, these are circuit-level factors and function-level factors. Each of these factors affects the power consumption of processors uniquely. The circuit-level factors mainly deal with the raw materials the circuitry of the processor was fabricated with, while the function-level factors deal with physical/structural design of the processors. The following subsections give further discussions on these.

3.1. Circuit level factors

Processors are essentially products of Complementary Metal Oxide Semiconductor (CMOS) technology and as such the circuit level factors which affect the power consumption of processors are static factors and dynamic factors.

The static factors occur when the circuit enters a standby mode where leakage current occurs due to static dissipation. The leakage current is characterized by five components viz: gate tunneling, gate-induced drain leakage, reverse-biased pn junction current, sub-threshold leakage, and punch-through effect [51–56]. For any CMOS-based device, the static power P_{static}consumption is defined as [57–59]:

$$P_{static} = V_{cc} \times I_{leakage} \tag{1}$$

where $V_{cc}$is the supply voltage and $I_{leakage}$is the sum of leakage currents in the device.

The dynamic factors occur due to power consumption when charging and discharging the output node capacitance of a transistor due to switching activity. There are two components in dynamic power consumption - switching power $P_{switching}$ due to charging and discharging of load capacitance, and short-circuit power P_{sc}due to nonzero input waveforms rise and fall time. These components are

related mathematically as [60–62]:

$$P_{dynamic} = P_{switching} + P_{sc} \tag{2}$$

The switching power consumption occurs as a result of the charging and discharging of the load capacitance, which is connected to the circuit. It is mathematically expressed as [63–67]:

$$P_{switching} = \alpha C_L V_{dd}^2 f_{clk} \tag{3}$$

where $\alpha$is the switching activity factor and it is estimated as the sum of the effective power-consuming voltage transitions per clock cycle, $C_L$is the load capacitance which is the sum of the device capacitance and interconnect capacitance, V_{dd} is the supply voltage, and f_{clk} is the switching frequency of the circuit.

The short-circuit power consumption occurs during output transition when there is a DC path between the supply and ground. Mathematically, it is expressed as [63,66]:

$$P_{sc} = I_{sc} V_{dd} = \frac{\beta}{12}(V_{dd} - 2V_t)^3 \frac{\tau}{T} \tag{4}$$

where I_{sc} is the short-circuit current, V_{dd} is the supply voltage, V_t is the threshold voltage, β is the gain factor of the transistor, T is the period, and τ is the input transition time.

3.2. Function level factors

Two main components make up the function level factors in the power consumption of a processor, they are: the datapath, and the cache. Other factors which contribute to the function level factors but are outside of the processor are the bus and main memory; these factors will not be discussed as they are outside the scope of the current discussion.

A datapath is a functional unit that performs data processing operations on input data using arithmetic logic unit, buses, multiplexers, and registers [68–71]. These components are either implemented as combinational circuits or sequential circuits in the datapath. As a result of this, switching power consumption is implicit during data operations by the datapath. Owing to the cocktail of arrangements and interconnections between these components, which are determined primarily by the complexity of the datapath itself, there is no one-stop expression that fully characterizes the power consumption in a datapath. Despite this, several attempts have been made in literature towards minimizing the power consumption in a datapath. Some authors explored clock gating, dual edge-triggered clock, and system latches to optimize power consumption in the datapath [72–74]. Using a technique called bit-slice activation, the lack of a high level of entropy in data streams was exploited in the reduction of power consumption in all implicit and explicit storage components of a typical superscalar datapath [75]. MOVER (Multiple Operating Voltage Energy Reduction) technique has been used in the minimization of datapath energy consumption through the use of multiple voltage supply sources; the technique finds a minimum voltage for the entire datapath from multiple sources of supply voltages [76].

The cache is a memory that stores frequently accessed data and instructions by the CPU; this bridges the capability gap between the main memory and the CPU [77,78]. The overall system data processing ability and instruction execution speed rely heavily on the performance of the cache. The cache has three functional parts - a data section, status information, and a directory store [79]. Using these parts, the cache performs a lot of data processing operations, and similar to the datapath, significant switching power consumption occurs. Also, owing to the different design approaches and implementation strategies, a single expression that completely characterizes the switching power consumption by the cache cannot be formulated. However, several attempts have been made in literature by different authors to minimize switching power consumption in caches. A data filter cache (DFC) has been integrated with a rapid address computation technique which reduced the impact of misses and significantly improved energy savings [80,81]. A dynamic data compression approach in secondary cache and a gated-V_{dd} control per cache block has been applied in achieving a reduction in the static power consumption and energy consumption by secondary caches [82].

Having looked at the factors that are responsible for power consumption which leads to the energy consumption by processors, the next section will examine the techniques by which the power and energy consumption of processors can be estimated.

4. Power and energy consumption estimation techniques in processors

The power consumption of a processor is a function of the energy consumption of the processor since power is the rate at which energy is consumed, and as stated in Section 3A, the power consumption is the sum of static power and dynamic power. A well-designed processor which optimally manages its power consumption is likely to be energy efficient, and this is a desirable property for any processor which will be involved in IoT applications and systems.

4.1. Transistor level power and energy consumption estimation

The transistor level is the lowest level for processor power and energy consumption estimation. It deals directly with the very nature of the semiconductor materials used in the fabrication of the transistor. These materials could be PMOS (p-type metal oxide semiconductor- constructed using a p-type source and drain and an n-type substrate), NMOS (n-type metal oxide semiconductor-constructed using an n-type source and drain and a p-type substrate), CMOS (complementary metal-oxide-semiconductor- constructed

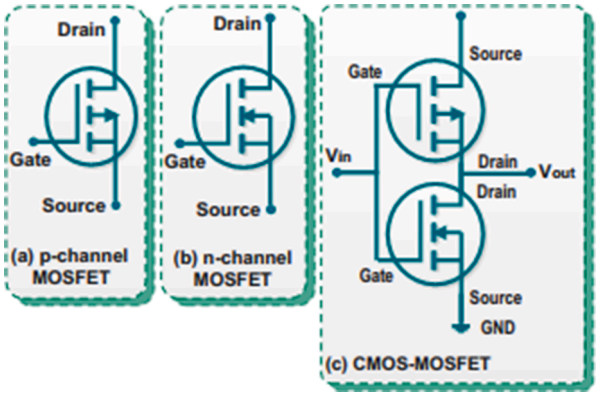


Fig. 3. Types of FETs fabricated using p-channel, n-channel, and CMOS MOSFETs.

using a combination of PMOS and NMOS) [83]. Fig. 3 shows the types of transistors called metal oxide semiconductor field-effect transistors (MOSFETs) fabricated using these materials.

In practice, the CMOS MOSFET plays a dominant role in integrated circuit and processor designs because a CMOS circuit has very fast transitions between the low and high states, it has the lowest power consumption, and most importantly, the output of a CMOS circuit makes full voltage oscillation between the low and high rail; this is not so with PMOS and NMOS circuits. Thus, a CMOS circuit output gives a fully digital 0 and digital 1. For these reasons, the power consumption relationships in this section apply to CMOS MOSFETS.

For CMOS MOSFETS, the relationships in (1)-(4) subsist in the determination of the power consumption at the transistor level of abstraction. In the case of energy consumption, there are two components involved i.e. static power in (1) and dynamic power in (2); these are static energy $E_{stat}$and dynamic energy E_{dyn}[84]. The static energy consumption is the result of the total leakage current (consisting of drain leakage, junction leakage, and gate leakage), and static current (consisting of DC bias current) [84]. For any given moment of time$t$: $(t > 0)$, the static energy consumption can be expressed as:

$$E_{stat}(t) = \int_0^t V_{DD}(I_{leak} + I_{DC})d\tau = V_{DD}(I_{leak} + I_{DC})t \tag{5}$$

where $V_{DD}$is the supply voltage, $I_{leak}$is the leakage current, and $I_{DC}$is the static current.

The dynamic energy which is a result of the relationships in (3) and (4) can be obtained by modeling the circuit in Fig. 3c as a capacitor $C_L$being charged by V_{DD}through a circuit with resistance R; the expression is represented as [84–87]:

$$E_{dyn} = C_L V_{DD}^2 \tag{6}$$

When half of the energy is consumed by R and half is saved by C_L, the relationship in (6) becomes:

$$E_{dyn} = \frac{C_L V_{DD}^2}{2} \tag{7}$$

Combining (5) and (7), the total energy consumed at the transistor level of abstraction is:

$$E_{tot} = E_{stat} + E_{dyn} + V_{DD}(I_{leak} + I_{DC})t + \frac{C_L V_{DD}^2}{2} \tag{8}$$

4.2. Gate level power and energy consumption estimation

This level of abstraction deals with the power and energy consumption of each logic gate in the CMOS logic circuit under consideration. For each gate, the power consumption is expressed as [88]:

$$P_{gate} = P_{cap} + P_{sc} + P_{leakage} \tag{9}$$

where $P_{sc}$and $P_{leakage}$are the short circuit power consumption and leakage power consumption respectively and previously discussed in Section 3A. $P_{cap}$is the capacitive power, and its consumption can happen during complete digital transitions and incomplete digital transitions (glitches - generated by colliding events caused by two or more input transitions) [88]. During complete transitions, P_{cap} is expressed as [88,89]:

$$P_{cap} = N(T)\frac{1}{2T}C_{in}V_{DD}^2 \tag{10}$$

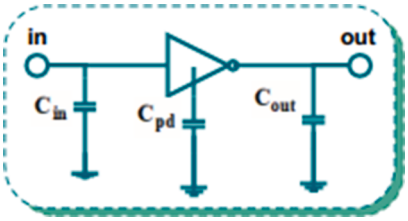


Fig. 4. Inverter circuit with three capacitances.

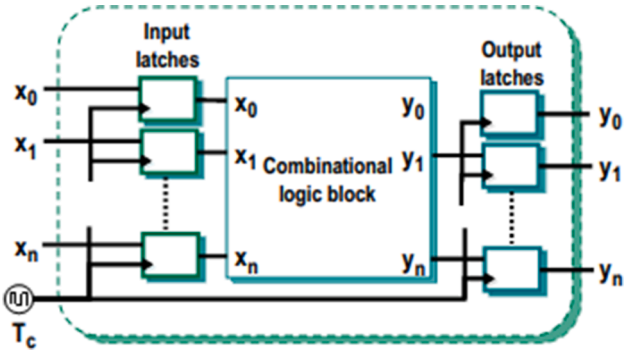


Fig. 5. A sequential design with an embedded combinational circuit.

where $N(T)$is the total complete transitions in the time interval T, $C_{in}$is the input capacitance. When a glitch occurs, $P_{cap}$is expressed as [88]:

$$P_{cap} = \frac{1}{2T} C_{in} V_{DD} \sum_{j=1}^{M(T)} \Delta V_j \tag{11}$$

where $M(T)$ is the total number of glitching transitions in the time interval T and ΔV_j is the peak voltage whose measurement is from the initial voltage value of the glitch transition j.

To estimate the energy consumed by a logic gate during the transition (switching activity), three capacitances are usually considered: the input capacitance C_{in}, power consumption capacitance $C_{pd}$of the gate, and capacitance due to connecting wires C_{out}. Consider an inverter shown in Fig. 4 [89] as an example.

For a single transition, the energy consumed can be estimated as [89]:

$$E_{transition} = \frac{1}{2}\left(C_{in} + C_{pd} + C_{out}\right) V_{DD}^2 \tag{12}$$

A statistical approach can also be applied in estimating the power consumption at the gate level of abstraction. Consider the combinational circuit embedded in a synchronous sequential design in Fig. 5 [90].

If the probability of a signal $P_s(x)$at a node $x$is defined as the average fraction of clock cycles for which x has a steady-state high logic, and $P_t(x)$is the transition probability at a node $x$and is defined as the average fraction of clock cycles for which $x$has a steady-state and its value is different from its initial value, then the power dissipated is [90]:

$$P_{av} = \frac{1}{2T_c} V_{dd}^2 \sum_{i=1}^{n} C_i P_t(x_i) \tag{13}$$

where $T_c$is the clock period, $C_i$is the total capacitance at node x_i, and $n$is the total number of nodes in the circuit. It should be noted that the nodes are the inputs/outputs of the latches, and the latches are logic gates.

In terms of the input patterns and state vectors, the power consumption in (13) can be expressed as [91]:

$$P = \frac{1}{2T_c} V_{dd}^2 \sum_{i=1}^{N_g} C_i \mathbf{n}_i(\mathbf{V}_1, \mathbf{S}_1, \mathbf{V}_2, \mathbf{S}_2) \tag{14}$$

where $N_g$is the number of gates in the circuit, $\mathbf{V}_1$is the present input pattern, $\mathbf{S}_1$is the present state vector, $\mathbf{V}_2$is the next input pattern, $\mathbf{S}_2$is the next state vector, and $\mathbf{n}_i$is the number of transitions at the node i.

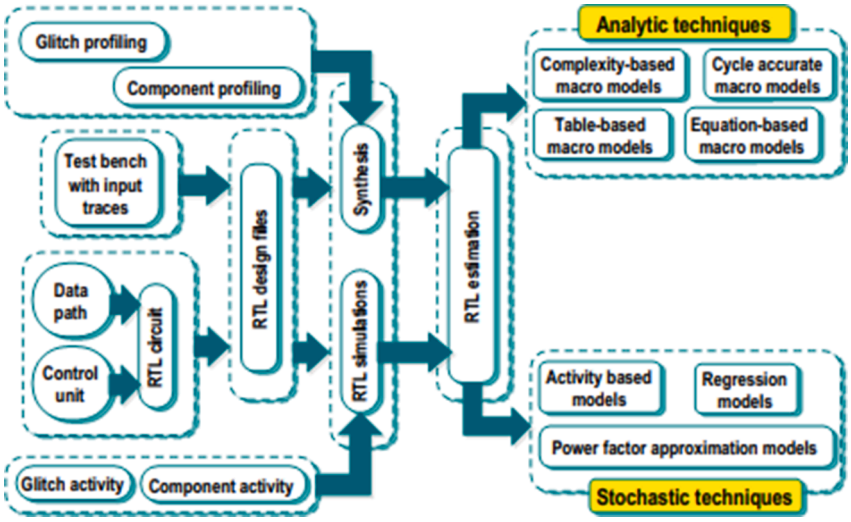


Fig. 6. RTL power estimation process and techniques.

4.3. Register-transfer-level (RTL) power and energy consumption estimation

Efficient estimation of power consumption for processor functions can be achieved with a good level of precision using the RTL level of abstraction. It is the view of the authors that a possible approach that can be used in this level of abstraction is shown in Fig. 6; it characterizes processor designs as an instantiation of pre-designed macroblocks which include flip-flops, multiplexers, arithmetic operators, and registers; this approach is a modification to the previous work by [92]. In this approach, glitch profiling and activity estimation are done concurrently with component profiling and activity estimation. The approach begins with the control unit and datapath design of the processor according to specifications; these are fused together as shown to produce the processor RTL circuit which executes the specified operations the processor is expected to perform. To ensure the processor operates correctly before RTL estimation, test benches with the necessary input traces are developed to verify the functional accuracy of the RTL circuit by instantiating the design, generating stimulus waveform, and reference outputs [93,94].

The combination of the RTL circuit and the test bench forms the RTL design files from which synthesis and RTL simulation are performed [95,96]. During synthesis, glitch profiling and component profiling are performed - these profiles make it possible to identify performance issues like the occurrence of glitches when components change state and the capacitance at component nodes. The RTL simulation provides information like clock speed, glitch duration, number of switching activities by components, and the frequency of glitches.

The output from the synthesis and RTL simulation is used as the primary drivers for the RTL estimation stage. As shown in Fig. 6, RTL power estimation can be done either using analytical, or stochastic techniques [97–99]. There are seven broad categories of these techniques in literature, and they are itemized and discussed as indicated in the following points.

1) Complexity-based models rely on the correlation that exists between chip architecture complexity and gate equivalents which specify the reference gates required for function implementation. Power consumption can then be estimated as a product of the approximate number of gates and the average power dissipated by each gate [100–102]. Eq. (15) shows this relationship [101, 103]:

$$P = \sum_{i\in(fns)} GE_i\left(E_{typ} + C_L^i V_{dd}^2\right) f . A_{\text{int}}^i \tag{15}$$

where GE_i is the block i gate equivalent circuit, $E_{typ}$is the average of the power dissipated by a gate, C_L is the average capacitance load, which includes fan-out and wiring, f is the clock frequency, and A_{int} is the percentage of gates switching in the block.

2) Cycle accurate macro models are linear functions that describe the statistical relationship between the power consumption of a vector pair and the characteristic values of the vector, and it is expressed in (16) as [104]:

$$P = B_0 + B_1X_1 + B_2X_2 + \cdots + B_kX_k \tag{16}$$

where P is the power consumption, $B_0, B_1, \ldots, B_k$ are regression coefficients of the macro model, and $X_1, X_2, \ldots, X_k$ are the characteristic variables that are extracted from the input vector pair. The cycle-accurate models are also called Cycle Accurate Functional Descriptions (CAFDs) and are known for precise and efficient specification of the behavior of circuits in the context of the cycle of their operations [105,106]. The efficiency is achieved through the omission of the details of the internal structure of circuits.

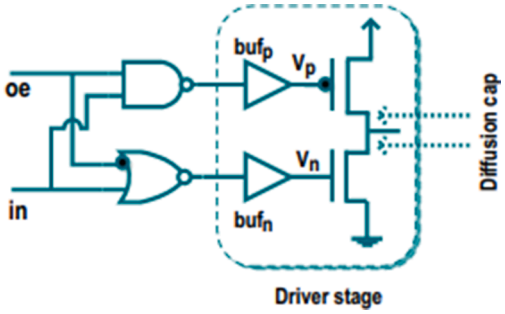


Fig. 7. Tristate driver circuit.

3) Table-based macro models involve the use of Look-Up-Tables (LUTs) for storing the estimates of equi-spaced discrete values which define the statistics of the input signal. To determine the distance between the discrete values, consider the following; let $\mathbf{n_p}$, $\mathbf{p^i_{in}}$, and $\mathbf{p^i_{LUT}}$ be respectively the number of parameter types with the i^{th}parameter extracted from input data. The distance *dist* is computed as [107,108]:

$$dist = \sqrt{\frac{1}{n_p}\sum_{i=0}^{n_p-1}\left(1-\frac{p^i_{in}}{p^i_{LUT}}\right)} \tag{17}$$

A known LUT in literature is the 3-D LUT whose power coefficients correspond to the following model [109–111]:

$$P = \mathrm{F}(P_{in}, D_{in}, D_{out}) \tag{18}$$

where$P_{in} = 1/N_{in}\sum_{i=1}^{N_{in}} Prob[in_i(t) = 1]$ is the input probability average, $D_{in} = 1/N_{in}\sum_{i=1}^{N_{in}} Prob[in_i(t) \neq in_i(t^+)]$the input transition average, and $D_{out} = 1/N_{out}\sum_{i=1}^{N_{out}} Prob[out_i(t) \neq out_i(t^+)]$the output transition average. These three parameters $\mathrm{P_{in}}$, $\mathrm{D_{in}}$, and $\mathrm{D_{out}}$ have values between 0 and 1, and the function is stored in a LUT with entries for each of the values of the three parameters. This results in a 3-D LUT, and Fig. 6 [111,112] shows a type of such table used in the storage of the parameters.

4) The equation-based macro models are built on statistical methods like least square estimation to achieve model formulation, and a principal feature of these models is the derivation of fitting coefficients which compute the power average power consumption with a reasonable degree of accuracy. One major advantage of this technique is that the equation (also called template) used for estimating the power consumption (even though non-linear) is generally fixed and robust to act as a starting point for all circuits [98], and as indicated by [98,113], the generalized polynomial form of the template can be expressed as:

$$\widehat{P}_{avg} = c_0 + c_1P_{in} + c_2D_{in} + c_3SC_{in} + c_4D_{out} \tag{19}$$

where $c_i$are unknown coefficients which are determined through regression analysis, P_{in} the average of the input signal probability, $D_{in}$the input switching activity average, $SC_{in}$the input signal correlation coefficient average, and $D_{out}$the output zero delay switching activity average. As indicated by [113], a quadratic form of (19) which gives better accuracy is expressed as:

$$\begin{aligned}\widehat{P}_{avg} &= c_o + c_1P_{in} + c_2D_{in} + c_3SC_{in} + c_4D_{out}\\ &+c_5P_{in}D_{in} + c_6P_{in}SC_{in} + c_7P_{in}D_{out}\\ &+c_8D_{in}SC_{in} + c_9D_{in}D_{out} + c_{10}SC_{in}D_{out}\\ &+c_{11}P^2_{in} + c_{12}D^2_{in} + c_{13}SC^2_{in} + c_{14}D^2_{out}\end{aligned} \tag{20}$$

5) Activity-based models rely on entropy from the field of information theory for the measurement of the average activity i.e. switching activity in the circuit under observation [114–116]. As shown in (13), power consumption is proportional to the product of transitional probability and capacitance; this makes it possible for entropy to measure the computational activity of the circuit. There are three basic methods for developing activity-based models- a probability approach, a regression approach, and the use of high-level tools capable of generating Switching Activity Interchange Format (SAIF) files.

For the probability approach, consider the Tristate circuit driver shown in Fig. 7 [117]; it can be observed that $V_p = \overline{oe.in}$ and $V_n = oe.\overline{in}$, hence the probability of switching activities for both $V_p$and V_n are:

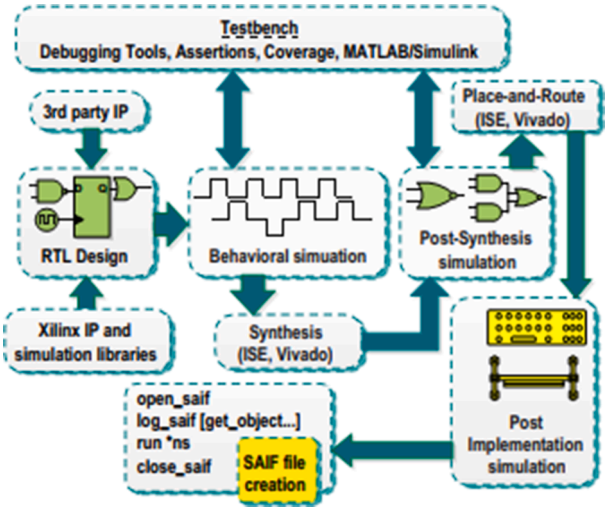


Fig. 8. SAIF file creation through Xilinx design flow.

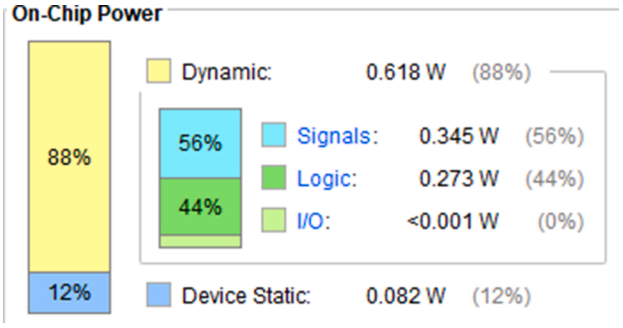


Fig. 9. Dynamic and Static power estimated from a SAIF file.

$$sw(V_p) = prob(oe = 1 \to 1, in = 0 \to 1) + prob(oe = 1 \to 1, in = 1 \to 0) + prob(oe = 1 \to 0, in = 1 \to x) + prob(oe = 0 \to 1, in = x \to 1) \tag{21}$$

$$sw(V_n) = prob(oe = 1 \to 1, in = 0 \to 1) + prob(oe = 1 \to 1, in = 1 \to 0) + prob(oe = 1 \to 0, in = 0 \to x) + prob(oe = 0 \to 1, in = x \to 0) \tag{22}$$

From where the average power consumption for every cycle of operation T is expressed as:

$$P_{av} = \frac{1}{T}\left[sw(V_p)C_{eff,buff_p} + sw(V_n)C_{eff,buff_n}\right]V_{dd}^2 \tag{23}$$

and the energy consumption per cycle is expressed as:

$$E = \left[sw(V_p)C_{eff,buff_p} + sw(V_n)C_{eff,buff_n}\right]V_{dd}^2 \tag{24}$$

The regression approach involves the expression of power consumption as a model in the following form [118]:

$$p_{kj} = \alpha + \beta_1 x_{1j} + \beta_2 x_{2j} + \cdots + \beta_n x_{nj} + \beta_{11} x_{1j}^2 + \beta_{22} x_{2j}^2 + \cdots + \beta_{12} x_{1j} x_{2j} + \cdots + \beta_{123} x_{1j} x_{2j} x_{3j} + \cdots \tag{25}$$

where p_{kj} is the power consumption in a module k, and x_{ij} is the toggle density of the signal i in a window j, α and β are parameters to be trained. Each variable in (25) from the perspective of activity-based modeling is the representation of certain switching activity in the circuit with its coefficient being its effective capacitance.

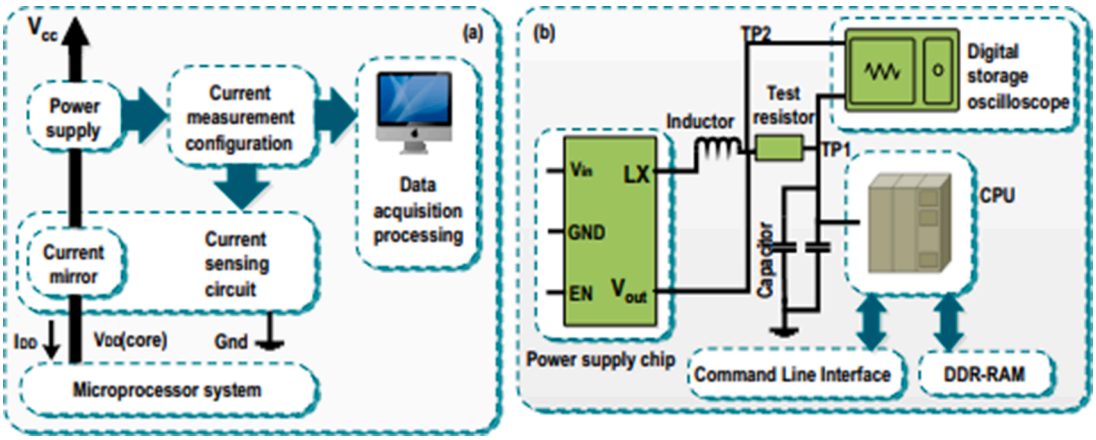


Fig. 10. Generalized circuits for physical measurements.

The use of high-level tools makes it possible to perform an activity-based power estimation for a design through the generation of a SAIF file which the tools enable. Fig. 8 [119] shows the flow process for the generation of a SAIF file using Xilinx ISE/VIVADO (high-level tools) as case studies. The authors modified this image obtained from the ALDEC website to include the SAIF file generation process block. The SAIF file enables Xilinx VIVADO to estimate the dynamic and static power utilization for a given design [120–122]. Fig. 9 shows a typical power utilization estimated by Xilinx VIVADO through SAIF files.

1) The Power Factor Approximation (PFA) technique makes a distinction between the effects of technology-independent and dependent factors in the estimation of power consumption [104,123–125]. It does this by using an experimentally determined weighting factor called power factor to derive a model for the average power consumption of a module over a range of designs. The technique also takes into account, the dependency on gate activities- this makes it stochastic, as gate activities are random processes. A key feature of PFA is its ability to explore algorithmic and architectural constraints at an early stage of a VLSI design [125]. The average power consumption using PFA is expressed as [124,125]:

$$P_{avg} = kGf \tag{26}$$

where G is the number of logic elements, f is the activation frequency, and $k = \frac{\sum_{i=1}^{G} A_{f_i} C_{L_i} V_{dd}^2}{G} + \frac{\sum_{i=1}^{G} A f_i V_{dd} \int_0^T I_{sc_i}(t)dt}{G}$, with A_f being the average number of switching per cycle.

2) Regression models perform RTL power estimation by correlating the input-output (I/O) activity with the average power; the average power is characterized as a function of the I/O activities and fitting parameters [126,127]. The fitting parameters are determined by the analysis of every component in the high-level design through simulation with pseudorandom data and fitting a multivariable regression curve to the power consumption results obtained. A regression model as proposed by [128] evaluates power consumption based on I/O activity which is represented as a vector of I/O Boolean variables $i = (i_1, i_2, \ldots, i_n)$and $o = (o_1, o_2, \ldots, o_m)$. If a value of 1 is observed whenever there is a transition on a corresponding I/O signal, then the power consumption can be expressed as:

$$p = c_0 + c_1 i_1 + c_2 i_2 + \cdots + c_n i_n + c_{n+1} o_1 + c_{n+2} o_2 + \cdots + c_{n+m} o_m \tag{27}$$

where $c = (c_0, c_1, \ldots, c_{n+m})$ are characteristically determined fitting coefficients. If s is the sample size of the data collected during characterization, and $\mathbf{X}$ is an $s \times (n+m+1)$ Boolean matrix such that $\mathbf{x}^k = (1, i_1^k, i_2^k, \ldots, i_n^k, o_1^k, o_2^k, \ldots, o_m^k)$then the expression in (27) can be written as:

$$\mathbf{p} = \mathbf{Xc} \tag{28}$$

This translates to:

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix} = \begin{bmatrix} i_{10} & i_{11} & i_{12} & \cdots & i_{1n} & o_{11} & o_{12} & \cdots & o_{1m} \\ i_{20} & i_{21} & i_{22} & \cdots & i_{2n} & o_{21} & o_{22} & \cdots & o_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i_{k0} & i_{k1} & i_{k2} & \cdots & i_{kn} & o_{k1} & o_{k2} & \cdots & o_{km} \end{bmatrix} \begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_n & c_{n+1} & c_{n+2} & \cdots & c_{n+m} \end{bmatrix}^T \tag{29}$$

Given the fact that in any logic circuit control signals have a strong influence on the behavior of logic units because they select different modes of operation for the circuit, then any of the regression equations in (29) will be selected by the control signals.

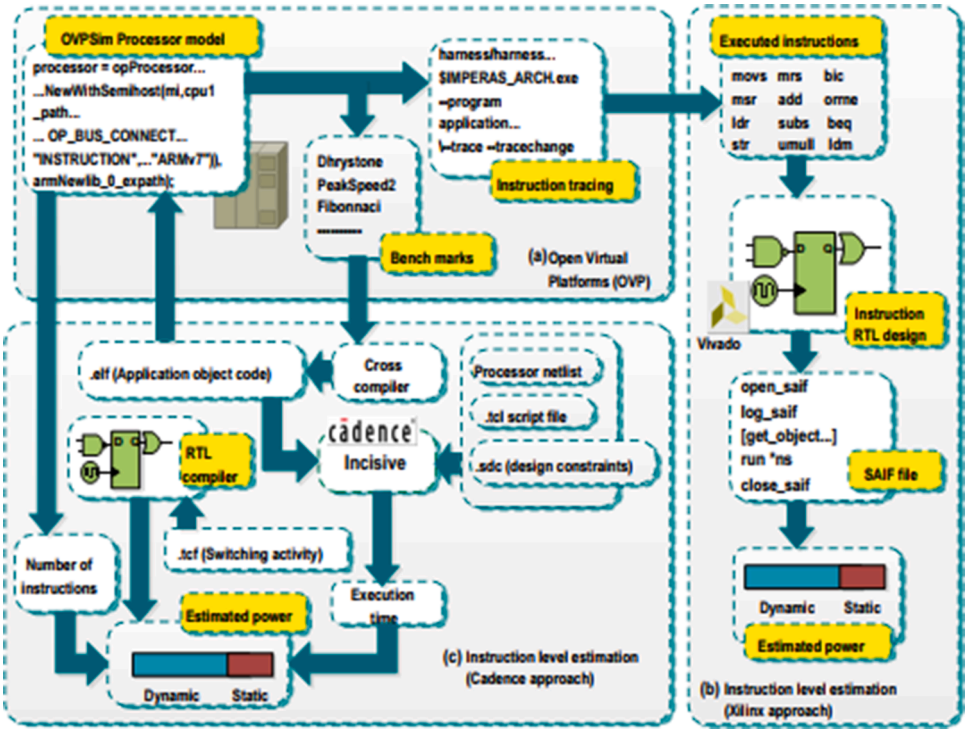


Fig. 11. Low-level simulation approach to instruction power/energy consumption.

4.4. Instruction level power and energy consumption estimation

This level of abstraction also called Instruction Level Power Analysis (ILPA) estimates the cost of executing each instruction in a program; the instructions are usually written in assembly code. The technique uses a low-level simulation of physical measurements in estimating the power and energy consumption of each instruction from the instruction set of an architecture [129].

Using the physical measurements approach for the estimation of instruction power and energy consumption, the technique requires that each assembly instruction is executed several times in a loop to mitigate the effects of branch instructions. During the loop, the current drawn by the processor as it executes the instruction is measured, and through the application of proper timing and signature analysis, the power and energy consumption can be estimated. There are two generalized approaches to physical measurement as depicted in Fig. 10 [130–134].

The configuration in Fig. 10a [131,132,134] involves the insertion of a current sensing circuit on the processor's power supply line, and the corresponding voltage drop across the sensing circuit causes the flow of current which is measured and subjected to timing and signature analysis. In the configuration in Fig. 10(b), the series resistor between the power supply and the CPU is used for power measurements and estimation, while the oscilloscope is used for the measurement of the instantaneous power. The DDR-RAM stores the instructions whose power is to be estimated, while the command-line interface is for loading the instruction into the CPU and for retrieving estimated power [130,135].

The low-level simulation approach involves an interaction between a cocktail of tools, which includes OVP (Open Virtual Platforms), and RTL design compilers for the design of the instructions under investigation; Fig. 11 shows such arrangement from two major developers - Xilinx and Cadence.

In Fig. 11a, the OVPSim models the processor whose instructions' power consumption is to be estimated [136–138]. Using the Xilinx approach as shown in Fig. 11b, the modeled processor then executes a specified benchmark, and the instructions used in the execution of the benchmark are traced. An RTL design is performed using the Xilinx VIVADO based on the model of the processor, and a SAIF is generated following the steps in Fig. 8, from which the dynamic and static powers are estimated [120–122]. Using the Cadence approach as shown in Fig. 11c, a cross-compilation is performed on a specified benchmark from which an application code is generated and run on the model of the processor using OVPSim; the application code also serves as an input to the Cadence Incisive simulation tool alongside a processor netlist (which contains the RTL specification of the processor), and a .tcl file (containing commands for performing simulations), and .sdc file (containing design constraints) [139–141]. The simulation tool outputs the execution time, and a .tcf file (containing switching activity). The number of instructions determined by OVPSim, the execution time, and the RTL compiled file is used in estimating the power consumption as indicated.

At instruction level, given a program*P*, its energy consumption can be estimated by the following relationship [130,142–145]:

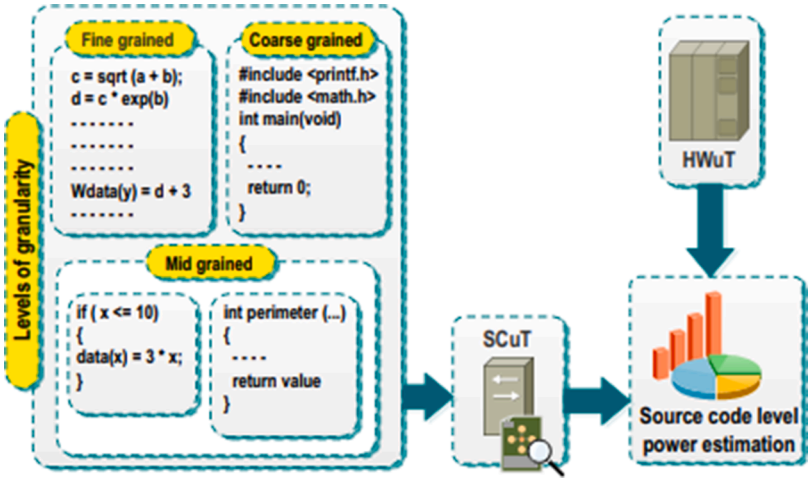


Fig. 12. Types of SCuT in source code level power estimation.

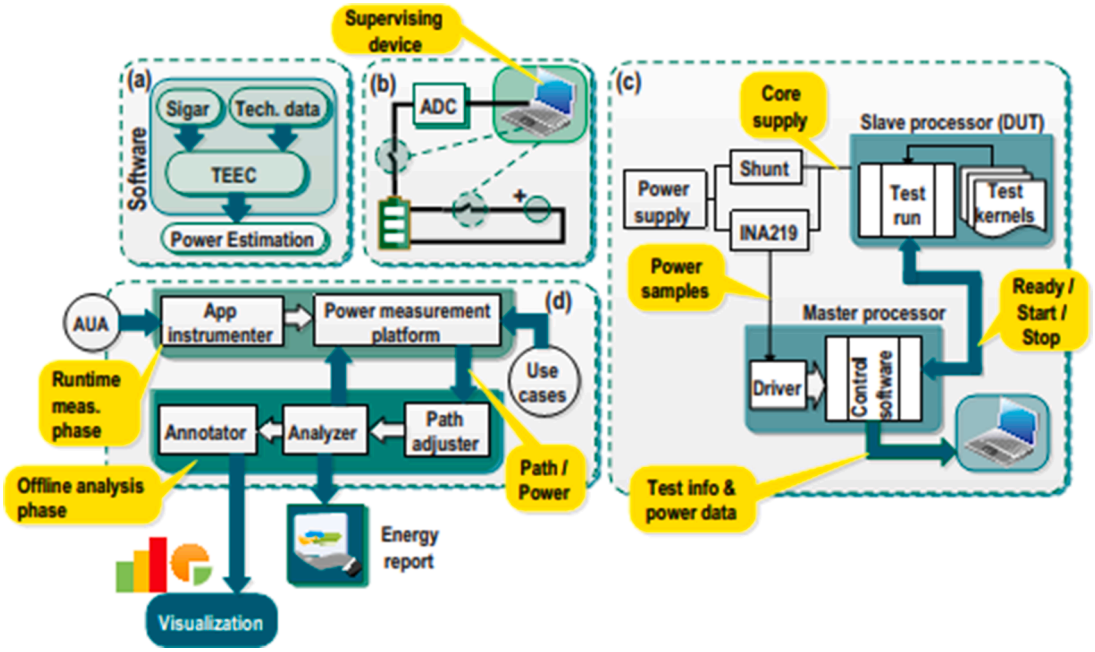


Fig. 13. Types of approaches to source code level power consumption estimation.

$$E_p = \sum_i (B_i \times N_i) + \sum_{i,j} (o_{i,j} \times N_{i,j}) + \sum_k E_k \tag{30}$$

where B_i is the base cost of instruction i weighted by the number of times it is executed N_i; $o_{i,j}$ is the circuit state overhead for the pair of consecutive instructions (i,j) weighted by the number of times it is executed $N_{i,j}$; E_k is the energy contribution of inter-instruction effects, and k is the stalls and cache misses which occur during program execution.

4.5. Source code level power and energy consumption estimation

This level of power and energy consumption is a complex technique that comprises two principal components - a source code component, which we call Source Code under Test (SCuT), and a hardware component, which we call Hardware under Test (HWuT). The combination of these two components as shown in Fig. 12 makes it possible to perform source code level power estimation. There are three possible states (fine-grained, mid-grained, and coarse-grained) for the SCuT, which can be used for the power estimation as indicated in Fig. 12 [146–148].

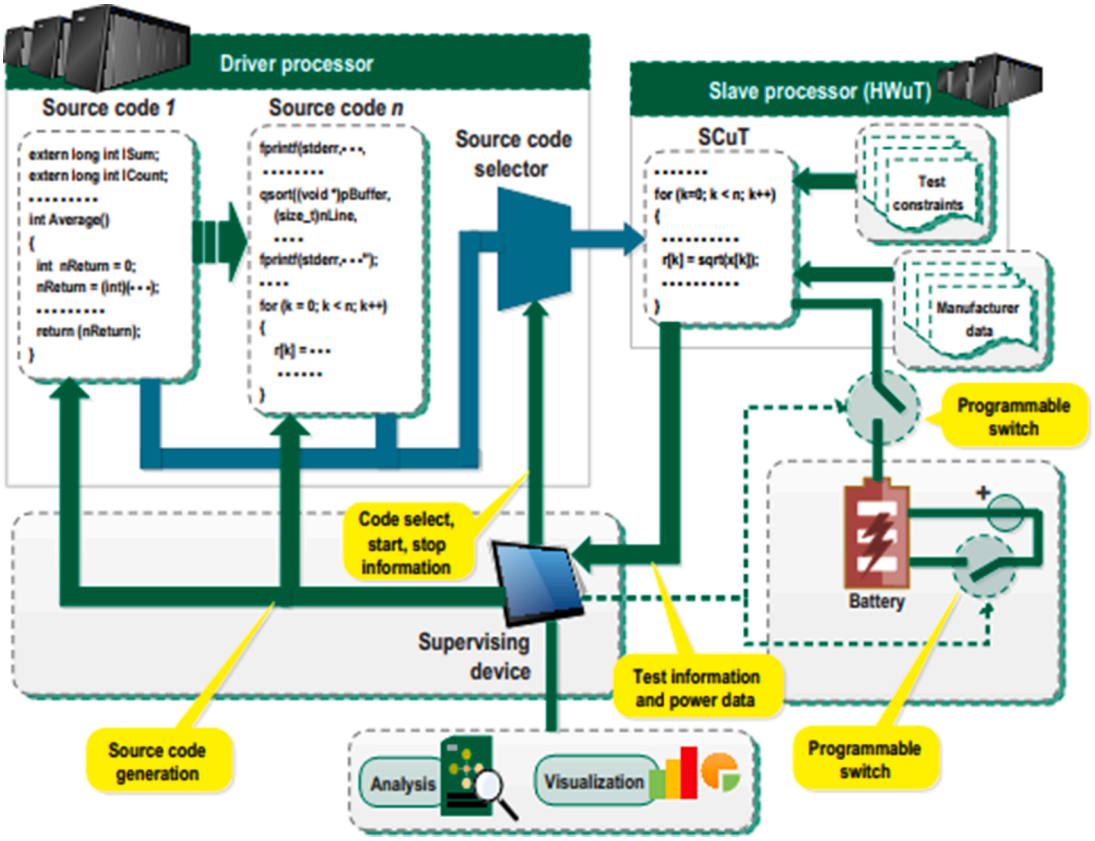


Fig. 14. A generalized approach for source code power and energy consumption estimation.

The fine-grained state corresponds to distinct lines of code under test, the mid-grained corresponds to functions or code segments under test, and the coarse-grained refers to complete programs under test [146,148]. An extensive investigation into an array of the literature reveals that different techniques have been proposed by several authors on source-level power and energy consumption estimation [146,149–156]; Fig. 13 shows four such techniques which generally cover all the variants of the types approach used by these authors. Using the approach in Fig. 13(a), the authors in [150] used a tool called TEEC (Tool to Estimate Energy Consumption) which had two inputs- Sigar library (which fetches information about CPU usage in terms of the number of cores) and percentage usage of each process and the technical data from the manufacturer in determining parts of a source code which has the highest power consumption. The authors in [146] used the approach in Fig. 13(b) where system times are recorded during a test run by a supervising device; the change in battery level and complete battery discharge are the basis of the system times. The power consumed by a source code is derived from the total time taken to completely discharge the battery during the test run of the source code. The concept of a master processor and slave processor- also called Device under Test (DUT) shown in Fig. 13(c) was used by the authors in [151] to perform source code power estimation. The master processor in their technique consisted of an xmprofile control software, which loads the slave processor with test source codes and then measures the power consumed by the slave processor in executing the source code. A runtime measurement phase and an offline analysis phase shown in Fig. 13(d) were used by the authors in [148] to perform source code power consumption estimation. In the runtime phase, the source code - called Application under Analysis (AUA) serves as an input into an App Instrumenter, which uses a path profiling technique in guiding the insertion of probes into the source code in order to capture the stamps and path traversal information. With a set of use cases, the power measurement platform records power samples from the source code. In the offline analysis phase, static analysis of paths and power samples modification are performed by the path adjuster. The energy consumed by each source code line is determined by the analyzer through regression analysis.

Through the aggregation of the techniques in Fig. 13, we propose what we call a generalized approach for source code level power and energy consumption estimation as depicted in Fig. 14. It is envisaged to act as a reconfigurable framework that can serve as a basis for the source code level power and energy consumption.

The idea shown in Fig. 14 involves a driver processor and a slave processor - Hardware under Test (HWuT), which is battery-powered. Different source codes are generated by a supervising device and loaded to the driver processor. The supervising device then selects which of the source codes gets loaded to the HWuT (flexibility is the reason for this approach). The source code which gets loaded to the HWuT is called Source Code under Test (SCuT), and it is executed by the slave processor under different test constraints as required and manufacturer data of the processor for which the source code will be deployed. The HWuT records the amount of time required to execute the SCuT and the energy expended by the battery during the execution of the SCuT; the combination of these can be

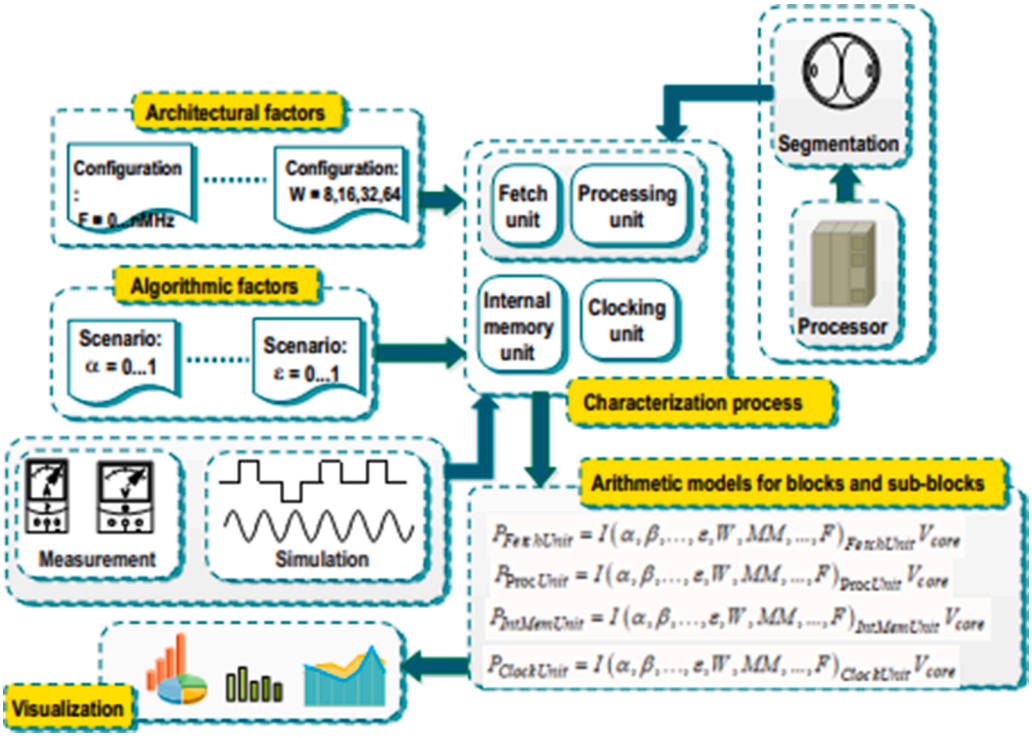


Fig. 15. Function level power and energy consumption estimation.

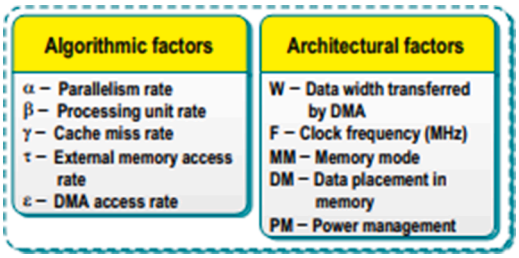


Fig. 16. Factors in function level power and energy consumption estimation.

used in determining the power consumed since power is the rate at which energy is consumed [152].

4.6. Function level power and energy consumption estimation

This level of power and energy estimation involves the segmentation of the architecture of the processor into different blocks and sub-blocks. Through simulations or measurements, an arithmetic model which determines the power consumption of each block can be derived as a dependency on parameters like clock frequency, degree of concurrency, rate of memory access, and register operations rate [157,158]. Fig. 15 shows the function level power estimation based on the descriptions in [157–159].

There are two parameters i.e. the algorithmic factors, and architectural factors as shown in Fig. 15 for the derivation of the power estimation model. The algorithmic factors are determined by the type of executed algorithm, and the architectural factors are determined by hardware performance metrics [159]. Fig. 16 shows the types of algorithmic and architectural factors typically used in function level power and energy consumption estimation [159].

4.7. System level power and energy consumption estimation

This is the highest level of abstraction in processor power estimation. Different methodologies for this level of abstraction have been proposed by several authors, and after a thorough literature review, three methodologies have been identified as outstanding because they represent a broad spectrum of the variances in other methodologies.

The first methodology is based on the use of an on-chip bus performance monitoring unit (PMU) which accurately estimates system-

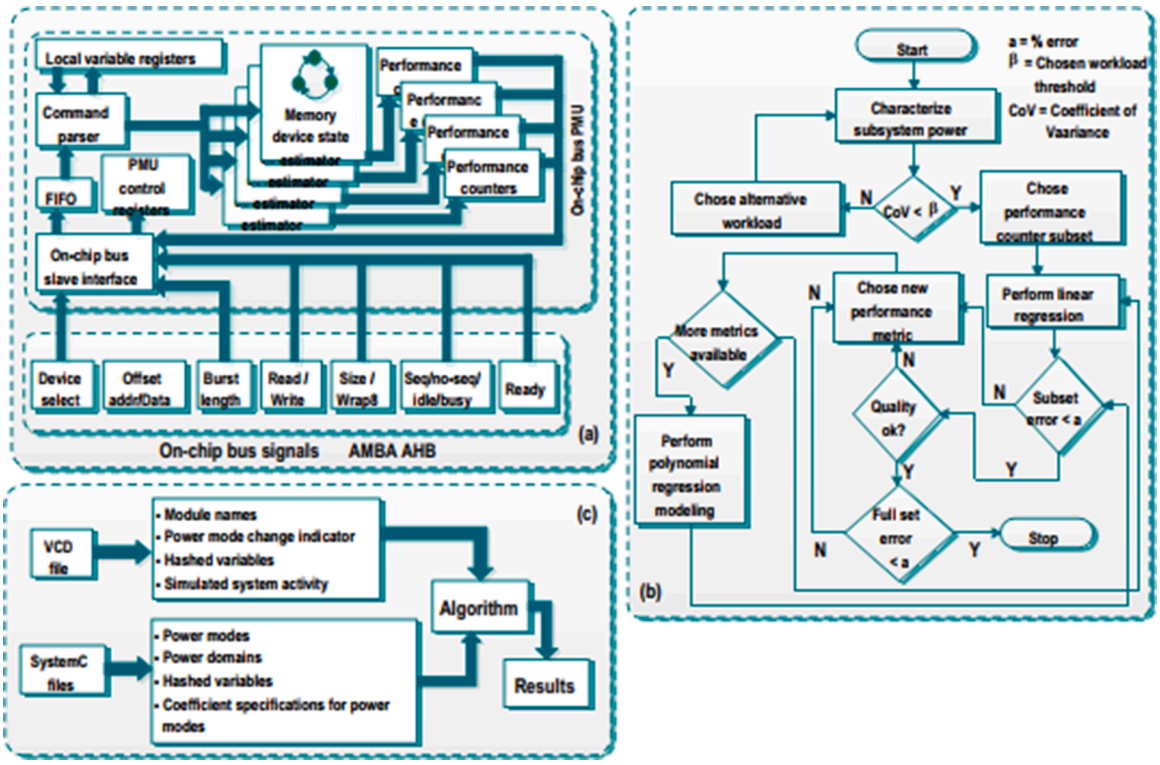


Fig. 17. Types of approaches for system-level power and energy consumption estimation.

level power consumption as a first-order linear power model through exchanged system-level activity information on the on-chip bus [160]. A major advantage this approach has over conventional PMUs and transaction-based PMUs is that the PMU is located on the on-chip bus where the exchange of information necessary for system-level power estimation is performed. Fig. 17(a) [160] shows the concept of this approach. It consists of a set of state machines and a slave interface. Upon the selection of a memory device, a command parser activates an associated state machine; this is followed by an appropriate encoding in the data bus and address bus for the appropriate memory devices. Other important signals which the PMU harnesses for power estimation includes instructions executes (c[IEX]), data dependencies (c[DDP]), instruction cache misses (c[ICM]), instruction TLB misses (c[ITM]), and data TLB misses (c[DTM]). A first-order linear model which constitutes these parameters can be expressed as [160]:

$$P_{cpu} = \alpha_1 c[IEX] + \alpha_2 c[DDP] + \alpha_3 c[ICM] + \alpha_4 c[ITM] + \alpha_5 c[DTM] \tag{31}$$

where $\alpha_1, \ldots, \alpha_5$ are the coefficients of power, while P_{static} is the processor static power consumption.

The second approach as shown in Fig. 16(b) is based on trickle-down power modeling which relies on the broad visibility of system-level events as seen by the processor [161]. With events that are local to the processor, this approach creates accurate performance counter-based models. A advantage major of this approach is that it does not require the creation of interfaces for multiple devices and subsystems whose performance counters have inconsistent APIs [161].

The operation of the trickle-down power model as indicated in Fig. 17(b) is such that it begins with a measurement of the system-level power through a subset of workloads [161]. A Coefficient of Variance (CoV) is checked to determine if it is greater than a specified threshold. For a given subsystem, performance counters are selected to measure performance events. Linear regression is then performed with the performance counter events as input variables, and subsystem power representing the output variable. The average error per sample is then determined using a subset of workloads.

The third approach as shown in Fig. 17(c) is built on a profiling activity using a modified form of Hamming distance computation to determine the number of signal activities [162]. The approach makes it possible to determine the numerical value of the bitwise switching activities of a component solely on simulation. The first step as shown is the processing of data from the Value Change Dump (VCD) and SystemC simulation files. An algorithm reads the files to calculate the Hamming distance, which is stored in a result file. To get the results, the algorithm concurrently searches the data in the SystemC file to determine the power status of a specified model while computing the Hamming distance [162]. For a determined model state, the product of the computed Hamming distance with a corresponding coefficient is determined and summed to the total energy consumption. This process is repeated till the end of the VCD file is reached.

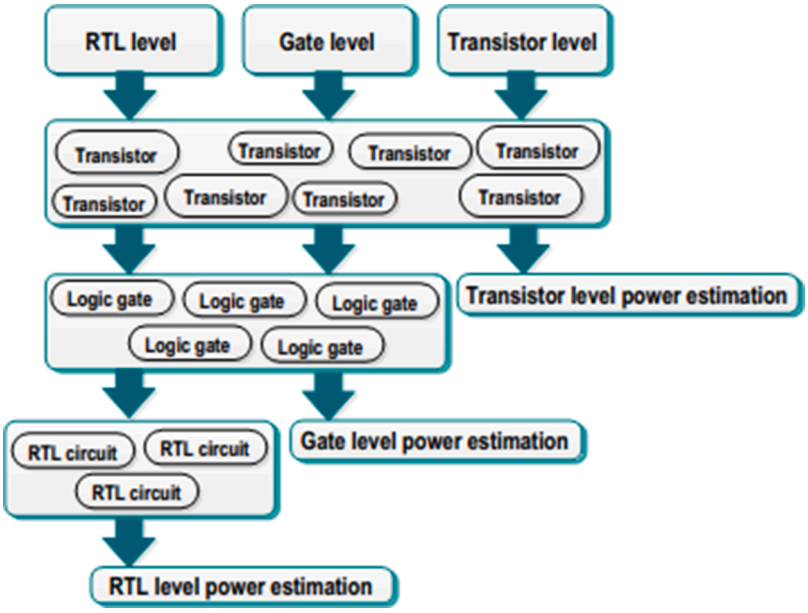


Fig. 18. Occurrence of overlap at lower level abstraction.

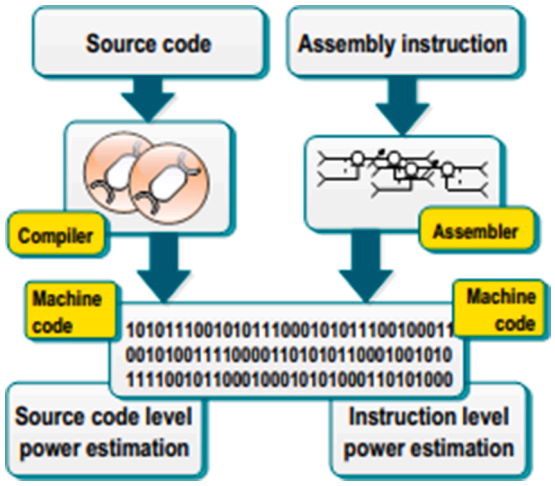


Fig. 19. Occurrence of overlap at middle level abstraction.

5. Overlapping factors and effect of power duty cycles in power and energy consumption estimation

The power estimation techniques as indicated in Fig. 2 show an increasing level of abstraction from the lower level of abstraction to the upper level of abstraction. As such, overlaps are bound to occur between power estimation techniques. A good knowledge of these overlaps makes it possible to give justification for the selection of power estimation factors and parameters as the level of abstraction increases. It is against this backdrop that this section presents these overlapping factors; this will be followed by a brief overview of the effect of power duty cycles on different abstraction levels.

At the lower level abstraction, logic gates are made of transistors, hence, power estimation at transistor level invariably translates into power estimation at gate level. The only difference is that while the transistor level relies on switching activity factors and switching frequency of the circuit, the gate level relies on complete and incomplete digital transitions, and glitches which occur during the operation of a circuit. Similarly, an overlap occurs between the RTL and the gate level. Because of the higher level of the RTL, power estimation determining factors (switching activity) at the transistor level and the gate level (digital transitions and glitches) are applied at the RTL level since it is a derivative of the two lower levels. Fig. 18 shows the overlap between these levels.

At middle level abstraction, an overlap can occur between source code level and instruction level estimation techniques because source codes (C, C++, and Rust) ultimately end up as machine code; the same is true of assembly instructions which define the

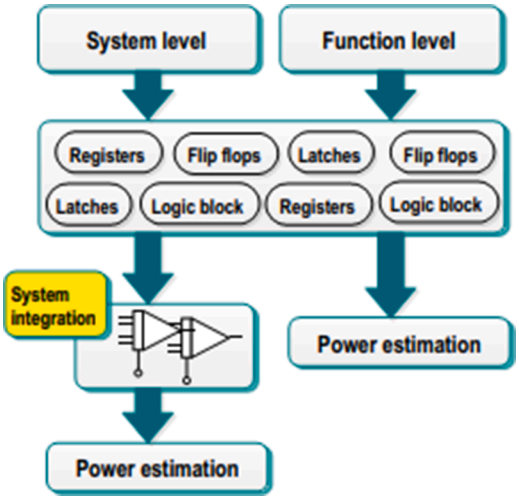


Fig. 20. Occurrence of overlap at upper level abstraction.

Table 2
Effect of power duty cycles on estimation techniques.

Estimation Technique	Effect of power duty cycles
Transistor level	PDC affects the switching power associated with the charging and discharging of the load capacitance as indicated in (3); this in turn affects the dynamic energy consumption in (6). If the PDC is configured such that the OFF power is significantly smaller than the ON power, then the mean power consumption is proportionally related to the PDC.
Gate level	PDC affects the number of transitions for any given time interval T as indicated in (10). The number of transitions is the switching activity whose rate is determined by the configuration of the PDC.
RTL level	The analytic and stochastic techniques are affected by parameters like the statistics of input signal, dependency on gate activities, and i/o activities. The rate at which these parameters occur are determined by the value of the PDC.
Instruction level	The number of instructions executed for any given time interval is indicated in (30). PDC determines this number in the sense that the higher the value of the PDC, the more number of instructions that will be executed for the time under consideration.
Source code level	It involves the use of tools like TEEC for fetching information about CPU usage of processes and threads; PDC plays a role in the sense that higher values of PDC indicate the presence of processes and threads in the source code that are performing intensive operations. Hence, constraining the PDC translates into putting a cap on maximum power consumption by the source code.
Function level	Algorithmic factors like DMA access rate, external memory access rate, processing unit rate depend on the configuration of the PDC. Higher values of PDC indicates that these factors will perform intensive operations which translate into higher power consumption rate.
System level	The number of signal activities determined using Hamming distance is a function of PDC. Hence, for higher PDC values which indicate higher power consumption rate, a high number of signal activities will be observed.

instruction level as they also ultimately end up as machine code. Hence, machine code is the ultimate bedrock which determines the rate of power consumption for middle level abstraction. However, the size of the machine code which determines the rate of power consumption is determined by the path (Source code level or Instruction level) from which it was arrived at. Fig. 19 shows the machine code overlap between the source code level and instruction level.

At the upper level abstraction in Fig. 2, an overlap in power estimation occurs between the function level and system level in the sense that while the function level focuses on registers and sub-blocks for power estimation, the same components are used by the system level for power estimation within the context of an integrated system. Thus while the function level gives differentiated power estimation using a given set of components and sub-blocks, the system level gives an integrated power estimation using the same components and sub-blocks. Fig. 20 shows the overlap between these two levels.

5.1. *Effect of power duty cycles*

For any given frequency value, a higher value of Power Duty Cycles (PDC) translates into higher power consumption [163]. It is an indicator of the activity rate in an electronic system [164]. PDC is directly proportional to the rate of power consumption for most configurations in an electronic system [165]; hence, as it increases, power consumptions rate also increases. From the foregoing, PDC has an effect on the factors which govern power and energy estimation techniques; the effect is determined by parameters which uniquely characterize each technique and described in Table 2.

Table 3
Strength and weakness of estimation techniques.

Technique	Strength	Weakness
Transistor level	• Most accurate • Has deep industry penetration	• For large designs, power estimation will take a long time to compute • It is very expensive
Gate level	• High accuracy • Libraries exist for predefined gates, which guarantees a short estimation time	• Computation complexity is high for large designs • For designs with large inputs, the presence of entropy in the input pattern due to glitches can degrade the accuracy of estimation
Register-transfer level	• Power can be determined using minimum inputs • Requires only architectural information of hardware	• If the estimation is statistics-based, accuracy is low • The latency for estimation is high when it is simulation-based
Instruction level	• High accuracy • Highly generic • It allows the complete isolation of the estimated power of an application from the hardware. This is ideal for developing energy-efficient IoT applications	• For large designs, power estimation has high latency • For the low-level simulation approach for estimation, the developer must have a high degree of proficiency in the use of an array of complex tools; this is usually difficult to get in practice.
Source code level	• The power consumption for different parts of a code can easily be determined • It is very flexible	• The complexity of estimation is inversely proportional to the code density • Errors in the source code directly affect the accuracy of estimation
Function level	• The segmentation of the processor into sub-blocks makes it possible to determine which part of the processor is exhibiting the most power consumption when an application is running • There is considerable flexibility in the degree of segmentation. This makes it convenient to perform estimation for exactly the task at hand	• Difficult to determine the set of input patterns when the power of a particular sub-block is to be estimated • A wrong value for the algorithmic or architectural factors will significantly degrade the accuracy of estimation
System-level	• The technique is easy to set up • It is efficient	• Industry penetration is low when compared with other techniques • The accuracy of estimation is subjective as there are divergent views about how best to view the system from a processor-perspective

6. Strength/weakness, technical challenges, and future direction of power and energy consumption estimation techniques

There is no technique, algorithm, or methodology in science or engineering that is foolproof. As a result of this, they all have their strengths/weaknesses, and technical challenges. It is against this backdrop that this section presents these issues for the techniques discussed in this paper.

6.1. Strength/weakness and technical challenges of power and energy consumption estimation techniques

Like any other technique or algorithm, the techniques discussed in Section IV have their strengths and weakness, as well as technical challenges associated with their realization. This section presents in tabular form, the strength/weakness of each technique, and the technical challenge of each technique. From the analysis of each of these techniques, Table 3 is derived, showing the advantages and disadvantages of each technique. From further literature review of the estimation techniques, Table 3 showing the technical challenges of each estimation technique is derived.

6.2. Future direction of power and energy consumption estimation techniques

One of the main issues affecting processor power and energy estimation techniques is the information gap between chipmakers and System & Application Design Engineers (SADE). This has had a significant effect on the ability of SADE to develop effective and accurate estimation techniques. Another factor is the seeming knowledge gap in tensor algebra by SADE. It is our candid opinion that tensor algebra if properly understood can provide an effective and efficient mechanism by which estimation techniques based on statistics can be developed. This is because with tensors, it is possible to represent all the necessary information for power and energy estimation for any number of predictor variables. If this is done, robust power and energy estimation models from a statistical approach can be developed.

As a way forward, it will be essential for the development of a concise framework that will bridge the gap between what chipmakers produce, and what SADE really needs to be effective in the development of power and energy estimation models.

With advancement in machine learning (ML) algorithms, a future direction for power and energy estimation technique based on

Table 4
Technical challenges of estimation techniques.

Technique	Technical challenge(s)
Transistor level	• Deriving a power distribution map of a given die by minimizing errors in estimation is still a challenge. This is succinctly represented by the expression $\mathbf{P} = \mathbf{AT}$where **P**and **T**are column vectors which represent power and temperature respectively, and **A**is the conductance matrix [142]. • At higher clock frequencies, metal becomes a significant part of delay in sub-quarter-micron design circuits [166,167].
Gate level	• It is still a challenge to develop an effective glitch filtering method capable of capturing the probability of states on a node of interest at more than one instance of time [100,168,169].
Register-transfer level	• Accurate measurement of digital circuits requires that all possible range of frequencies for which power consumption is inconsistently spread must be considered [170]. This is not possible in practice as demonstrated in (15) where $P = \sum_{i\in(fns)} GE_i(E_{typ} + C_L^i V_{dd}^2) f.A_{\text{int}}^i, \forall f$ and (26) where $P_{avg} = kGf, \forall f$
Instruction level	• The overhead cost of loop control like adds, jumps, and compares cannot be distinguished from the cost of executing the instructions of interest [171]; this has a significant effect on the aggregate accuracy of the technique especially for large loops.
Source code level	• The presence of components like pipelines, caches, branch prediction, and other speculative components makes it impractical to determine the upper bounds for the worst-case execution time (WCET) time problem of a given source code [172]. • As the clock frequency of the processor increases with advanced designs, achieving fast memory access becomes a challenge because of the increase in latency in communication between the processor and off-chip cache or memory [173]. This inevitably affects the accuracy of power consumption estimation.
Function level	• There is no one-size-fits-all sample size for the algorithmic parameters used in this technique; this is because the varying degree of the quality of the running codes will produce different sample sizes, which will likely have a significant effect on the accuracy of the technique based on the relationships in Fig. 15. It is therefore impractical to determine an optimal sample size because the idea of code quality is speculative [159].
System-level	• There is a widening gap between design automation tools and what is actually needed for system level power estimation [174]. • Many chip makers are still at the early stages of wrestling with very complex power management schemes [174].

statistics is that predictor variables for any estimation technique, which can be represented by tensors, can be effectively churned out by ML algorithms to produce an accurate prediction model. This approach has been made possible by the increasing role tensors are playing in ML operations [175–178].

7. Conclusion

Processor power and energy consumption remains one of the most important factors to be considered in the design and development of IoT solutions because the entire operations and control of IoT nodes and devices are performed by processors. There are different types of processors with different levels of performance for any given technological solution. As a result of this, different techniques have emerged for the estimation of the power and energy consumption of processors. It is against this backdrop that this paper reviewed different techniques which have been used for the estimation of processor power and energy consumption from the lowest abstraction level i.e. transistor level to the highest level of abstraction i.e. system level. The review focused on each technique in the context of their methodology and the required tools for the estimation of power and energy consumption. The effect of overlaps in this layered techniques and processor duty cycle values were equally discussed. Two Tables were provided in which the first showed the advantages and disadvantages of each technique; the second summarised the technical challenges of each technique.

Funding statement

This work was supported by TETFUND National Research Fund under Grant TETFUND/RD&D/NRF/STI/56/Vol.1.

Declaration of Competing Interest

None.

Data availability

No data was used for the research described in the article.

References

[1] G.A. Akpakwu, B.J. Silva, G.P. Hancke, A.M. Abu-Mahfouz, A survey on 5G networks for the Internet of Things: communication technologies and challenges, IEEE Access 6 (2018) 3619–3647.

[2] R. Hassan, F. Hassan, M.K. Hasan, A.H.M. Aman, A.S. Ahmed, Internet of Things and its applications: a comprehensive survey, MDPI - Symmetry 12 (1674) (2020) 2–29.
[3] A. Sinha, Study on IoT (Internet of Things) and its Applications, Int. J. Res. Appl. Sci. Eng. Technol. 8 (VI) (2020) 2533–2536.
[4] N. Sathiyanathan, S. Selvakumar, P. Selvaprasanth, A brief study on IoT applications, Int. J. Trend Sci. Res. Dev. 4 (2) (2020) 23–27.
[5] S. Pless and P. Torcellini, "Net zero energy buildings: a classification system based on renewable energy supply options," Colorado, 2010.
[6] S. Steers, "Vodafone report finds 5G, IoT could help cut CO2 emissions," Vodafone - 5G & IoT, 2021. [Online]. Available: https://mobile-magazine.com/5g-and-iot/vodafone-report-finds-5g-iot-could-help-cut-co2-emissions. [Accessed: 06-Jul-2021].
[7] S. O'Meara and Y. Ye, "Four research teams powering China's net-zero energy goal," Nat. Spotlight - China's net-zero ambitions, vol. 603, pp. 541–543, 2022.
[8] A.T. Salim, S.I. Basheer, N.R. Saadallah, A survey of using Internet of Things to enhance zero energy buildings, Texas J. Multidiscip. Stud. 1 (1) (2021) 205–213.
[9] Bouckaert et al, "Net zero by 2050: a roadmap for the global energy sector," Paris France, 2021.
[10] R.J. Cole, Net-zero and net-positive design: a question of value, Build. Res. Inf. 43 (1) (2015) 1–6.
[11] World Economic Forum, The Global Risks Report 2022, 17th ed., World Economic Forum, 2022.
[12] M. Stead, A. Gradinar, P. Coulton, J. Lindley, Edge of tomorrow: designing sustainable edge computing, in: Synergy - DRS International Conference, 2020, pp. 88–110.
[13] A. Finkel, "Smart zero: using advanced networks to accelerate progress towards net zero," 2022.
[14] P. Bellini, P. Nesi, G. Pantaleo, IoT-enabled smart cities: a review of concepts, frameworks and key technologies, MDPI - Appl. Sci. 12 (1607) (2022) 1–21.
[15] R.B. Vaidya, S. Kulkarni, V. Didore, Intelligent transportation system using IOT: a review, Int. J. Res. Trends Innov. 6 (9) (2021) 80–87.
[16] V. Arun, M. Poongothai, Implementation of IoT based intelligent transportation system, Asian J. Appl. Sci. Technol. 2 (2) (2018) 933–946.
[17] M. Visan, S.L. Negrea, F. Mone, Towards intelligent public transport systems in smart cities; collaborative decisions to be made, in: The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021) 199, 2022, pp. 1221–1228.
[18] B.R. Haverkort, A. Zimmermann, Smart industry: how ICT will change the game!,", Ieee Int. Comput. (2017) 8–10.
[19] G. Lampropoulos, K. Siakas, T. Anastasiadis, Internet of Things (IoT) in industry: contemporary application domains, innovative technologies and intelligent manufacturing, Int. J. Adv. Sci. Res. Eng. 4 (10) (2018) 109–118.
[20] T. Kalsoom, et al., Impact of IoT on manufacturing industry 4.0: a new triangular systematic review, MDPI - Sustain 13 (12506) (2021) 1–22.
[21] M.H. Abidi, M.K. Mohammed, H. Alkhalefah, Predictive maintenance planning for industry 4.0 using machine learning for sustainable manufacturing, MDPI - Sustain 14 (3387) (2022) 1–27.
[22] S. Nangia, S. Makkar, R. Hassan, IoT based predictive maintenance in manufacturing sector, in: International Conference on Innovative Computing and Communication, 2020, pp. 1–7.
[23] H. Yang, W. Lee, H. Lee, IoT smart home adoption: the importance of proper level automation, Hindawi - J. Sensors 2018 (2018) 1–12.
[24] L.W. Santoso, R. Lim, K. Trisnajaya, Smart home system using Internet of Things, J. Inf. Commun. Converg. Eng. 16 (1) (2018) 60–65.
[25] M. A. Tunc, E. Gures, and I. Shayea, "A survey on IoT smart healthcare: emerging technologies, applications, challenges, and future trends," *arXiv e-prints*, pp. 1–15, 2021.
[26] X. Yang, et al., Exploring emerging IoT technologies in smart health research: a knowledge graph analysis, BMC Med. Inform. Decis. Mak. 20 (260) (2020) 1–12.
[27] I. Keshta, AI-driven IoT for smart health care: Security and privacy issues, Elsevier -, Inform. Med. Unlocked 30 (2022) 1–7.
[28] G.J. Lakshmi, M. Ghonge, A.J. Obaid, Cloud based IoT smart healthcare system for remote patient monitoring, EAI Endorsed Trans. Pervasive Heal. Technol. 7 (28) (2021) 1–11.
[29] S. Katiyar, A. Farhana, Smart agriculture: the future of agriculture using AI and IoT, J. Comput. Sci. 17 (10) (2021) 984–999.
[30] G.V Dankan, P.M. Sandeep, M. Ramesha, M.K. Jayashre, S. Ansuman, Smart agriculture and smart farming using IoT technology, J. Phys. Conf. Ser. 2089 (012038) (2021).
[31] V.K. Quy, et al., IoT-enabled smart agriculture: architecture, applications, and challenges, MDPI - Appl. Sci. 12 (3396) (2022) 1–19.
[32] J. Yang, A. Sharma, R. Kumar, IoT-based framework for smart agriculture, Int. J. Agric. Environ. Inf. Syst. 12 (2) (2021) 1–14.
[33] A. Rehman, T. Saba, M. Kashif, S.M. Fati, S.A. Bahaj, H. Chaudhry, A revisit of Internet of Things technologies for monitoring and control strategies in smart agriculture, MDPI - Agron 12 (127) (2022) 1–21.
[34] D.B.S. Fernando, M. Sabarishwaran, R.R. Priya, S. Santhoshini, Smart agriculture monitoring system using IoT, Int. J. Sci. Res. Eng. Trends 6 (4) (2020) 2212–2216.
[35] A.E.P. Dany, X. Shuming, Agriculture monitoring system using smart and innovative farming: a real-time study, Int. J. Sci. Technol. Res. 8 (12) (2019) 1214–1219.
[36] J. Xu, B. Gu, G. Tian, Review of agricultural IoT technology, Artif. Intell. Agric. 6 (2022) 10–22.
[37] S.I. Hassan, M.M. Alam, U. Illahi, M.A. Al Ghamdi, S.H. Almotiri, M.M. Su'ud, A systematic review on monitoring and advanced control strategies in smart agriculture, IEEE Access 9 (2021) 32517–32548.
[38] H. Pal, S. Tripathi, A survey on IoT-based smart agriculture to reduce vegetable and fruit waste, J. Phys. Conf. Ser. 2273 (012009) (2022) 1–12.
[39] D. Thamaraiselvi, R.V Krishna, S. Hemateja, Smart and automated agricultural management system using IoT, Int. J. Innov. Technol. Explor. Eng. 11 (4) (2022) 49–55.
[40] E.S. Mohamed, A. Belal, S.K. Abd-Elmabod, M.A. El-Shirbeny, A. Gad, M.B. Zahran, Smart farming for improving agricultural management, J. Remote Sens. Sp. Sci. 24 (2021) 971–981.
[41] J. Haj-Yahya, A. Mendelson, Y.Ben Asher, A. Chattopadhyay, Power management of modern processors. Energy Efficient High Performance Processors: Recent Approaches for Designing Green High Performance Computing, Springer, Singapore, 2018, pp. 3–6.
[42] Y. Fan, J. Wu, S. Wang, energy efficiency of task allocation for embedded JPEG systems, Hindawi - Sci. World J 2014 (2014) 1–8.
[43] L. Ye, et al., The challenges and emerging technologies for low-power artificial intelligence IoT systems, IEEE Trans. Circuits Syst. I Regul. Pap. 68 (12) (2021) 4821–4834.
[44] S.S. Anjum, R.M. Noor, I. Ahmedy, M.H. Anisi, Energy optimization of sustainable Internet of Things (IoT) systems using an energy harvesting medium access protocol, IOP Conf. Ser. Earth Environ. Sci. 268 (012094) (2019) 1–7.
[45] V. Barot, R. Patel, Energy consumption optimization in Internet of Things applications: concept and techniques, Int. J. Adv. Sci. Technol. 29 (3) (2020) 9743–9751.
[46] F. Pereira, R. Correia, P. Pinho, S.I. Lopes, N.B. Carvalho, Challenges in resource-constrained IoT devices: energy and communication as critical success factors for future IoT deployment, MDPI - Sensors 20 (6420) (2020) 1–30.
[47] D. Airehrour, J. Gutiérrez, S.K. Ray, Greening and optimizing energy consumption of sensor nodes in the Internet of Things through energy harvesting: challenges and approaches, in: International Conference on Information Resources Management, 2016, pp. 1–13.
[48] M.A. Imran, A. Zoha, L. Zhang, Q.H. Abbasi, Grand challenges in IoT and sensor networks, Front. Commun. Networks (2020) 1–6.
[49] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, "Microprocessor optimizations for the Internet of Things: a survey," arXiv e-prints, pp. 1–14, 2018.
[50] Z. Sheng, D. Tian, V.C.M. Leung, Toward an energy and resource efficient Internet of Things: a design principle combining computation, communications, and protocols, IEEE Commun. Mag. 56 (7) (2018) 89–95.
[51] D. Helms, E. Schmidt, W. Nebel, Leakage in CMOS circuits – an introduction,, in: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation. PATMOS 2004. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2004, pp. 17–35.
[52] K. Roy, S. Mukhopadhyay, H. Mahmoodi-Meimand, Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits, Proc. IEEE 91 (2) (2003) 305–327.

[53] S.M. Pandey, P. Rawat, Calculation of leakage current in CMOS circuit design in DSM technology, Int. J. Comput. Appl. 155 (11) (2016) 22–26.
[54] A. Dixit, Transistor leakage mechanisms and power reduction techniques in CMOS VLSI design, Int. J. Adv. Res. Comput. Commun. Eng. 5 (3) (2016) 102–107.
[55] F. Fallah, M. Pedram, Standby and active leakage current control and minimization in CMOS VLSI circuits, IEICE Trans. Electron. 88–C (2005) 509–519.
[56] M. Drazdziulis, P. Larsson-Edefors, A gate leakage reduction strategy for future CMOS circuits, in: ESSCIRC 2004 - 29th European Solid-State Circuits Conference, 2003, pp. 317–320.
[57] A. Sarwar, CMOS power consumption and Cpd calculation, Texas Instrum. (1997) 1–12.
[58] N.S. Kim, et al., Leakage current: Moore's law meets static power, Computer (Long. Beach. Calif) 36 (12) (2003) 68–75.
[59] V. Venkatachalam, M. Franz, Power reduction techniques for microprocessor systems, ACM Comput. Surv. 37 (3) (2005) 195–237.
[60] N.B. Romli, K.N. Minhad, M.B.I. Reaz, M.S. Amin, An overview of power dissipation and control techniques in cmos technology, J. Eng. Sci. Technol. 10 (3) (2015) 364–382.
[61] J. Samanta, B.Prasad De, B. Bag, R.K. Maity, Comparative study for delay & power dissipation of CMOS inverter in UDSM range, Int. J. Soft Comput. Eng. 1 (6) (2012) 162–167.
[62] K. Kaur, A. Noor, Power estimation analysis for cmos cell structures, Int. J. Adv. Eng. Technol. 3 (2) (2012) 293–301.
[63] R.Y. Reetu, Dynamic power reduction of VLSI circuits: a review, Int. J. Adv. Res. Electron. Commun. Eng. 7 (3) (2018) 245–249.
[64] Y. Wang, D. Nörtershäuser, S.Le Masson, J. Menaud, Experimental characterization of variation in power consumption for processors of different generations, in: 15th IEEE International Conference on Green Computing and Communications, 2019, pp. 1–9.
[65] M. Kumar, Dynamic power dissipation analysis in CMOS VLSI circuit design with scaling down in technology, J. Act. Passiv. Electron. Devices 12 (2017) 55–61.
[66] A. Babu, Power optimization techniques at circuit and device level in digital CMOS VLSI – a review, Int. J. Eng. Res. Technol. 3 (11) (2014) 375–379.
[67] S. Parameswaran, H. Guo, Power consumption in CMOS combinational logic blocks at high frequencies, in: ASP-DAC '97: Asia and South Pacific Design Automation Conference, 1997, pp. 195–200.
[68] E.O. Hwang, Digital Logic and Microprocessor Design With VHDL, Brooks/Cole, New York, 2005.
[69] S.L. Harris, D.M. Harris, Digital Design and Computer Architecture: ARM Edition, Morgan Kaufmann, Waltham, USA, 2016.
[70] J. Davis, R. Reese, Finite State Machine Datapath Design, Optimization, and Implementation, Morgan & Claypool, New York, 2008.
[71] J.E. Stine, Digital Computer Arithmetic Datapath Design Using Verilog HDL, Springer Science+Business Media, New York, 2004.
[72] T. Subhashini, M. Kamaraju, Power optimized datapath units of hybrid embedded core architecture using clock gating technique, Int. J. VLSI Des. Commun. Syst. 6 (6) (2015) 33–43.
[73] M. Kamaraju, K. Lal Kishore, A.V.N. Tilak, Power optimized ALU for efficient datapath, Int. J. Comput. Appl. 11 (11) (2010) 39–43.
[74] R.K. Megalingam, T. Shekhil Hassan, P. Vivek, A. Mohan, M. Tanmay Rao, Power consumption reduction in CPU datapath using a novel clocking scheme, in: 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 529–533.
[75] D. Ponomarev, G. Kucuk, and K. Ghose, "Power reduction in superscalar datapaths through dynamic bit-slice activation," in *Innovative Architecture for Future Generation High-Performance Processors and Systems*, 2001, pp. 16–24.
[76] M.C. Johnson, K. Roy, Datapath scheduling with multiple supply voltages and level converters, Purdue (1996).
[77] R. Stacpoole, T. Jamil, Cache memories, IEEE Potentials 19 (2) (2000) 24–29.
[78] E. Conrad, S. Misenar, J. Feldman, Eleventh Hour CISSP: Study Guide, Syngress, Cambridge United States, 2017.
[79] A.N. Sloss, D. Symes, C. Wright, ARM System Developer's Guide Designing and Optimizing System Software, Morgan Kaufmann, San Francisco, 2004.
[80] A. Bardizbanyan, M. Själander, D. Whalley, P. Larsson-Edefors, Towards a performance- and energy-efficient data filter cache, in: Proceedings of the 10th Workshop on Optimizations for DSP and Embedded Systems, 2013, pp. 21–28.
[81] J. Lee, S. Kim, Filter data cache: an energy-efficient small L0 data cache architecture driven by miss cost reduction, IEEE Trans. Comput. 64 (7) (2015) 1927–1939.
[82] K. Tanaka, T. Kawahara, Leakage energy reduction in cache memory by data compression, ACM Sigarch Comput. Archit. News 35 (5) (2007) 17–24.
[83] P. Horowitz, W. Hill, The Art of Electronics, 2nd ed., Cambridge University Press, Cambridge UK, 1989.
[84] I. Ratković, N. Bežanić, O.S. Ünsal, A. Cristal, V. Milutinović, An overview of architecture-level power- and energy-efficient design techniques, Adv. Comput. 98 (2015) 1–57.
[85] S. Kang, Y. Leblebici, CMOS Digital Intergrated Circuit: Analysis and Design, 2nd ed, McGraw Hill, New York, 2003.
[86] N. Xiang-Jie, L. Hua, Lower power design for UHF RF CMOS circuits based on the power consumption acuity, Math. Probl. Eng. 2014 (2014) 1–8.
[87] B. Jacob, S.W. Ng, D.T. Wang, S. Rodriguez, Memory Systems: Cache, DRAM, Disk, Morgan Kaufmann, Burlington, 2008.
[88] L. Kruse, D. Rabe, W. Nebel, VHDL power simulator: power analysis at gate-level, in: C.D. Kloos, E. Cerny (Eds.), Hardware Description Languages and their Applications, Springer, Boston, MA, 1997.
[89] V. Chereja, A. Potarniche, S. Ranga, B.S. Kirei, M.D. Topa, Power dissipation estimation of CMOS digital circuits at the gate level in VHDL, in: 2018 International Symposium on Electronics and Telecommunications (ISETC), 2018, pp. 1–4.
[90] F.N. Najm, Estimating power dissipation in VLSI circuits, EEE Circuits Devices Mag 10 (4) (1994) 11–19.
[91] Y. Yuan, C. Teng, S. Kang, Statistical estimation of average power dissipation in sequential circuits, in: Proceedings of the 34th Design Automation Conference, 1997, pp. 377–382.
[92] A. Raghunathan, S. Dey, N.K. Jha, Register-transfer level estimation techniques for switching activity and power consumption, in: IEEE International Conference on Computer-Aided Design, 1996, pp. 158–165.
[93] Intel, "VHDL test bench file (.vht) definition," 2017. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_vht.htm. [Accessed: 30-Dec-2021].
[94] J. Deepak, "Testbenches in VHDL – a complete guide with steps," technobyte, 2020. [Online]. Available: https://technobyte.org/testbench-vhdl-types-examples-steps/. [Accessed: 30-Dec-2021].
[95] N. Kularatna, Electronic Circuit Design: From Concept to Implementation, CRC Press, Boca Raton, 2008.
[96] P.P. Chu, RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, John Wiley & Sons, Hoboken, New Jersey, 2006.
[97] C. Piguet, Low-Power CMOS Circuits: Technology, Logic Design and CAD Tools, CRC Press, Boca Raton, 2006.
[98] S. Gupta, F.N. Najm, Analytical models for RTL power estimation of combinational and sequential circuits, IEEE Trans. Comput. Des. Integr. Circuits Syst. 19 (7) (2000) 808–814.
[99] M. Pedram, Advanced power estimation techniques, in: J. Mermet, W. Nebel (Eds.), Low Power Design in Deep Submicron Technology. NATO ASI Series (Series E: Applied Sciences), Kluwer Academic Publishers, Boston, MA, 1997, pp. 179–201.
[100] Y. Nasser, J. Lorandel, J. Prevotet, M. Hélard, RTL to transistor level power modelling and estimation techniques for FPGA and ASIC: a survey, IEEE Trans. Comput. Des. Integr. Circuits Syst. 40 (3) (2021) 479–493.
[101] S.R. Nesset, RTL Power Estimation Flow and Its Use in Power Optimization, Norwegian University of Science and Technology, 2018.
[102] L. Chelini, "Power estimation for DSP components for fiber-optic communication systems," Chalmers University Of Technology | University Of Gothenburg, 2017.
[103] K. Kirsch, K. Muller-Glaser, K. Neusinger, Estimating essential design characteristics to support project planning for ASIC design management, in: IEEE International Conference on Computer-Aided Design'91, 1991, pp. 148–151.
[104] Q. Wu, Q. Qiu, M. Pedram, C. Ding, Cycle-accurate macro-models for RT-level power analysis, IEEE Trans. Very Large Scale Integr. Syst. 6 (4) (1998) 520–528.
[105] L. Zhong, S. Ravi, A. Raghunathan, N.K. Jha, RTL-aware cycle-accurate functional power estimation, IEEE Trans. Comput. Des. Integr. Circuits Syst. 25 (10) (2006) 2103–2117.
[106] L. Zhong, S. Ravi, A. Raghunathan, N.K. Jha, Power estimation for cycle-accurate functional descriptions of hardware, in: IEEE/ACM International Conference on Computer Aided Design, 2004, pp. 668–675.

[107] H. Kawauchi, I. Taniguchi, M. Fukui, A new approach for accurate RTL power macro-modeling, J. Semicond. Technol. Sci. 10 (1) (2010) 11–19.
[108] H. Kawauchi, M. Tsuzuki, I. Taniguchi, M. Fukui, An accurate RTL power estimation considering power library unevenness, in: IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 2618–2621.
[109] M. Barocci, L. Benini, A. Bogliolo, B. Ricco, and G. De Micheli, "Lookup table power macro-models for behavioral library components," in *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999, pp. 173–181.
[110] A. Bogliolo, R. Corgnati, E. Macii, M. Poncino, Parameterized RTL power models for soft macros, IEEE Trans. Very Large Scale Integr. Syst. 9 (6) (2001) 880–887.
[111] R. Corgnati, E. Macii, M. Poncino, Clustered table-based macromodels for RTL power estimation, in: IEEE Ninth Great Lakes Symposium on VLSI, 1999, pp. 354–357.
[112] S. Gupta, F.N. Najm, Power macromodeling for high level power estimation, in: IEEE Proceedings of the 34th Design Automation Conference, 1997, pp. 365–370.
[113] S. Gupta and F. N. Najm, "Analytical model for high level power modeling of combinational and sequential circuits," in *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999, pp. 164–172.
[114] P.P. Sotiriadis, A.P. Chandrakasan, A bus energy model for deep submicron technology, IEEE Trans. Very Large Scale Integr. Syst. 10 (3) (2002) 341–350.
[115] L. Benini, A. Macii, E. Macii, M. Poncino, R. Scarsi, Architectures and synthesis algorithms for power-efficient bus interfaces, IEEE Trans. Comput. AIDED Des. Integr. Circuits Syst. 19 (9) (2000) 969–980.
[116] I. Lee, et al., PowerViP: SoC power estimation framework at transaction level, in: IEEE Asia and South Pacific Conference on Design Automation, 2006, p. 8.
[117] C. Hsieh, M. Pedram, Architectural power optimization by bus splitting, in: Design, Automation, and Test in Europe Conference and Exhibition, 2000, pp. 612–616.
[118] D. Kim, "*FPGA-Accelerated Evaluation and Verification of RTL Designs*," University of California, Berkeley, 2019.
[119] ALDEC, "Xilinx design flow," FPGA Design: FPGA Vendors Support, 2022. [Online]. Available: https://www.aldec.com/en/solutions/fpga_design/fpga_vendors_support/xilinx–xilinx-fpga-design-flow. [Accessed: 07-Jan-2022].
[120] Xilinx, "Vivado design suite tutorial: power analysis and optimization," UG997 (v2021.2), 2021. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_1/ug997-vivado-power-analysis-optimization-tutorial.pdf. [Accessed: 07-Jan-2022].
[121] Xilinx, "Vivado design suite user guide: logic simulation," UG900 (v2021.2), 2021. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2021_2/ug900-vivado-logic-simulation.pdf. [Accessed: 07-Jan-2022].
[122] Xilinx, "Xilinx power estimator user guide," UG440 (v2021.2), 2021. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manuals/xilinx2021_2/ug440-xilinx-power-estimator.pdf. [Accessed: 07-Jan-2022].
[123] M.D. Grammatikakis, G. Kornaros, M. Coppola, Power-aware multicore SoC and NoC design, in: M. Hubner, J. Becker (Eds.), Multiprocessor System-on-Chip: Hardware Design and Tool Integration, Springer, New York, 2011, pp. 167–193.
[124] R. Amirtharajah, "EEC 216 lecture #2: metrics and logic level power estimation." Davis, 2008.
[125] P.M. Chau, S.R. Powell, Power dissipation of VLSI array processing systems, J. VLSI Signal Process. 4 (1992) 199–212.
[126] E. Macii, M. Pedram, F. Somenzi, High-level power modeling, estimation, and optimization, IEEE Trans. Comput. Des. Integr. Circuits Syst. 17 (11) (1998) 1061–1079.
[127] A. Bogliolo, L. Benini, Node sampling: a robust RTL power modeling approach, in: IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers, 1998, pp. 461–467.
[128] L. Benini, A. Bogliolo, M. Favalli, G. De Micheli, Regression models for behavioral power estimation, Integr. Comput. Aided. Eng. 5 (2) (1998) 95–106.
[129] H. Blume, D. Becker, L. Rotenberg, M. Botteck, J. Brakensiek, T.G. Noll, Hybrid functional- and instruction-level power modeling for embedded and heterogeneous processor architectures, Elsevier -, J. Syst. Archit. 53 (2007) 689–702.
[130] W. Wang, M. Zwolinski, An improved instruction-level power model for ARM11 microprocessor. High Performance Energy Efficient Embedded Systems (HIP3ES), 2013, pp. 1–7.
[131] S. Nikolaidis, T. Laopoulos, Instruction-level power consumption estimation of embedded processors for low-power applications, Elsevier -, Comput. Stand. Interfaces 24 (2002) 133–137.
[132] N. Kavvadias, P. Neofotistos, S. Nikolaidis, C.A. Kosmatopoulos, T. Laopoulos, Measurements analysis of the software-related power consumption in microprocessors, IEEE Trans. Instrum. Meas. 53 (4) (2004) 1106–1112.
[133] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, T. Laopoulos, Energy consumption estimation in embedded systems, IEEE Trans. Instrum. Meas. 57 (4) (2008) 797–804.
[134] M. Bazzaz, M. Salehi, A. Ejlali, An accurate instruction-level energy estimation model and tool for embedded systems, IEEE Trans. Instrum. Meas. 62 (7) (2013) 1927–1934.
[135] V.A. Kulkarni, G.R. Udupi, Instruction level power consumption estimation – issues and review, J. Multidiscip. Eng. Sci. Technol. 4 (2) (2017) 6776–6781.
[136] A. Wicaksana, C.M. Tang, Virtual prototyping platform for multiprocessor system-on-chip hardware/software co-design and co-verification. Computer and Information Science, Springer, Cham, 2018, pp. 93–108.
[137] Imperas Software Limited, "OVP guide to using processor models," Open Virtual Platforms, 2021. [Online]. Available: https://www.imperas.com/sites/default/files/documents/OVP_Guide_To_Using_Processor_Models.pdf. [Accessed: 22-Jan-2022].
[138] Imperas Software Limited, "OVP processor modeling guide," Open Virtual Platforms, 2021. [Online]. Available: https://www.ovpworld.org/documents/OVP_Processor_Modeling_Guide.pdf. [Accessed: 22-Jan-2022].
[139] H. Vankani, A. Venkatramanan, and D. S. Ha, "Front end design using cadence tool - analyze and compile," Multifunctional Integrated Circuits and Systems Group, 2021. [Online]. Available: https://www.mics.ece.vt.edu/ICDesign/Tutorials/Cadence/dg_front_analyze.html. [Accessed: 22-Jan-2022].
[140] F. Rosa, L. Ost, T. Raupp, F. Moraes, R. Reis, Fast energy evaluation of embedded applications for many-core systems, in: 24th International Workshop on Power and Timing Modeling, Optimization and Simulation, 2014, pp. 1–6.
[141] G.S.P. Delicia, T. Bruckschloegl, P. Figuli, Bringing accuracy to open virtual platforms (OVP): a safari from high-level tools to low-level microarchitectures, in: International Conference on Innovations In Intelligent Instrumentation, Optimization And Signal Processing, 2013, pp. 22–27.
[142] H. Sultan, G. Ananthanarayanan, S.R. Sarangi, Processor power estimation techniques: a survey, Int. J. High Perform. Syst. Archit. 5 (2) (2014) 93–114.
[143] T. Li, L.K. John, Run-time modeling and estimation of operating system power consumption, ACM Sigmetrics Perform. Eval. Rev. 31 (1) (2003) 160–171.
[144] V. TIWARI, S. MALIK, A. WOLFE, Instruction level power analysis and optimization of software, J. VLSI Signal Process. Syst. 13 (2–3) (1996) 223–238.
[145] M.E.A. Ibrahim, M. Rupp, H.A.H. Fahmy, A precise high-level power consumption model for embedded systems software, EURASIP J. Embed. Syst. 2011 (2011) 1–14.
[146] L. Ardito, R. Coppola, M. Morisio, M. Torchiano, Methodological guidelines for measuring energy consumption of software applications, Hindawi - Sci. Program. (2019) 1–16.
[147] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: IEEE 8th International Conference on SoftwareTesting,VerificationandValidation(ICST), 2015, pp. 1–12.
[148] D. Li, S. Hao, W.G.J. Halfond, R. Govindan, Calculating source line level energy information for android applications, in: International Symposium on Software Testing and Analysis, 2013, pp. 78–89.
[149] H.H.M. Hassan, A.S. Moussa, I. Farag, Performance vs. power and energy consumption: impact of coding style and compiler, Int. J. Adv. Comput. Sci. Appl. 8 (12) (2017) 132–142.
[150] H. Acar, G.I. Alptekin, J. Gelas, P. Ghodous, The impact of source code in software on power consumption, Int. J. Electron. Bus. Manag. 14 (2016) 42–52.
[151] U. Liqat et al., "Energy consumption analysis of programs based on XMOS ISA-level models," in *Logic-Based Program Synthesis and Transformation*, 2013, pp. 72–90.

[152] E. García-Martín, C.F. Rodrigues, G. Riley, H. Grahn, Estimation of energy consumption in machine learning, Elsevier -, J. Parallel Distrib. Comput. 134 (2019) 75–88.
[153] C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto, The impact of source code transformations on software power and energy consumption, J. Circuits, Syst. Comput. 11 (5) (2002) 477–502.
[154] D. Li, Z. Cui, C. Bai, Q. He, X. Yan, A tool for energy consumption monitoring and analysis of the android terminal, Hindawi - J. Electr. Comput. Eng. (2021) 1–11.
[155] H. Liu, F. Yan, S. Zhang, T. Xiao, J. Song, Source-level energy consumption estimation for cloud computing tasks, IEEE Access 6 (2018) 1321–1330.
[156] S. Hao, D. Li, W.G.J. Halfond, R. Govindan, Estimating mobile application energy consumption using program analysis, in: IEEE - 35th International Conference on Software Engineering, 2013, pp. 92–101.
[157] M.E.A. Ibrahim, M. Rupp, Power estimation methodology for VLIW digital signal processors, in: Proceedings of the Asilomar Conference on Signals, Systems, and Computers, 2008.
[158] H. Blume, M. Schneider, T.G. Noll, Power estimation on functional level for programmable processors, Adv. Radio Sci. 2 (2004) 215–219.
[159] J. Laurent, E. Senn, N. Julien, E. Martin, Functional level power analysis: an efficient approach for modeling the power consumption of complex processors, HAL - Open Sci (2004) 666, hal-000139.
[160] Y. Cho, Y. Kim, S. Park, N. Chang, System-level power estimation using an on-chip bus performance monitoring unit, in: 2008 IEEE/ACM International Conference on Computer-Aided Design, 2008, pp. 149–154.
[161] W.L. Bircher, L.K. John, Complete system power estimation using processor performance events, IEEE Trans. Comput. 61 (4) (2012) 563–577.
[162] J. Erdelyi, D. Macko, K. Jelemenska, PESL: system-level estimation of power-management effect on dynamic energy consumption, MDPI - Electron 9 (1313) (2020) 1–13.
[163] R. Rodriguez-Zurrunero, A. Araujo, M.M. Lowery, Methods for lowering the power consumption of OS-based adaptive deep brain stimulation controllers, MDPI - Sensors 2349 (2021) 1–17.
[164] L. Wanner, C. Apte, R. Balani, P. Gupta, M. Srivastava, Hardware variability-aware duty cycling for embedded sensors, IEEE Trans. Very Large Scale Integr. Syst. 21 (6) (2013) 1000–1012.
[165] J. Saraswat, P.P. Bhattacharya, Effect of duty cycle on energy consumption in wireless sensor networks, Int. J. Comput. Networks Commun 5 (1) (2013) 125–140.
[166] S. Goswami, S. Akashe, Performance and analysis of ultra deep sub micron technology using complementry metal oxide semiconductor inverter, Int. J. VLSI Des. Commun. Syst. 5 (6) (2014) 75–80.
[167] B. I. Abdulrazzaq, I. A. Halin, S. Kawahito, R. M. Sidek, S. Shafie, and N. A. Yunus, "A review on high-resolution CMOS delay lines: towards sub-picosecond jitter performance," Springerplus, vol. 5, no. 434, pp. 1–32, 2016.
[168] W. Fischer, B.M. Gammel, Masking at gate level in the presence of glitches, in: Cryptographic Hardware and Embedded Systems – CHES 2005, 3659, Springer, Berlin, Heidelberg, 2005, pp. 187–200.
[169] P. Kiaei et al., "Gate-level side-channel leakage assessment with architecture correlation analysis," arXiv e-prints, pp. 1–12, 2022.
[170] P. Bernardi, et al., Peak power estimation: a case study on CPU cores, in: 2012 IEEE 21st Asian Test Symposium, 2012, pp. 167–172.
[171] K.D. Cooper, L. Torczon, Introduction to optimization. Engineering a Compiler, 2nd ed., Morgan Kaufmann, Burlington, MA, 2012, p. 414.
[172] R. Wilhelm, et al., The worst-case execution time problem: overview of methods and survey of tools, ACM Trans. Embed. Comput. Syst. V (N) (2007) 1–47.
[173] M. Forsell, S. Nikula, J. Roivainen, V. Leppänen, J.L. Träf, Performance and programmability comparison of the thick control fow architecture and current multicore processors, J. Supercomput. 78 (2022) 3152–3183.
[174] A.S. Mutschler, Dealing with system-level power, Semiconduct. Eng. (2017) [Online]. Available: https://semiengineering.com/dealing-power-system-level/ [Accessed: 23-Oct-2022].
[175] J.A. Reyes, E.M. Stoudenmire, Multi-scale tensor network architecture for machine learning, Mach. Learn. Sci. Technol. 2 (035036) (2021) 1–14.
[176] T. Carter, "Enabling AI & machine learning: the role of tensor cores," Curtiss-Wright Defense Solutions, 2021. [Online]. Available: https://www.curtisswrightds.com/sites/default/files/2021-09/Enabling-AI-Machine-Learning-the-role-of-Tensor-Cores-White-Paper.pdf. [Accessed: 26-Oct-2022].
[177] R. Sengupta, S. Adhikary, I. Oseledets, and J. Biamonte, "Tensor networks in machine learning," *arXiv e-prints*, pp. 1–8, 2022.
[178] Y. Ren, D. Goldfarb, Tensor normal training for deep learning models, in: 35th Conference on Neural Information Processing Systems, 2021.