Low Power Scheduling for High Level Synthesis

Ukamaka Akumili Anaedu
Department of Electronic Engineering
Hochschule Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
email:Ukamaka-akumili.anaedu@stud.hshl.de

***Abstract*—By automatically converting high-level descriptions into hardware designs, high-level synthesis (HLS) is a crucial component in the design of contemporary digital systems. Low power scheduling strategies in HLS have attracted a great deal of attention as the demand for energy-efficient systems keeps growing. This abstract offers a thorough analysis of low power scheduling techniques for high-level synthesis. The primary objective of low power scheduling in HLS is to minimize power consumption while meeting performance constraints. Traditional scheduling approaches focused on optimizing performance metrics, such as latency and throughput, without explicitly considering power consumption. However, with the proliferation of battery-operated devices and the increasing awareness of energy conservation, power optimization has become a critical concern in HLS.**

***Index Terms*—Low power, High level synthesis, optimization, operation implementation, future directions.**

I. INTRODUCTION

A crucial tool for automating the design of sophisticated digital systems is high-level synthesis (HLS). HLS helps designers to quickly create and optimize digital circuits by converting high-level descriptions into hardware designs. Power optimization is becoming a crucial factor in the HLS flow due to the rising need for energy-efficient devices. Low power scheduling methods are essential for reducing power usage during synthesis while yet achieving performance goals. The goal of low power scheduling in HLS is to create a schedule of activities using the hardware resources that are readily available that reduces power consumption.According to [3] Traditional scheduling techniques didn't explicitly take into account power limits and instead prioritized performance indicators like throughput and latency. However, digital systems' ability to use less power has improved paramount due to factors such as limited battery life in portable devices and the rising concerns about environmental sustainability. Through careful management of the allocation of processes to hardware resources and the use of power-aware transformations, low power scheduling strategies seek to reduce power usage. These methods cover a variety of tactics, such as heuristic-based algorithms and methods for mathematical optimization. Heuristic-based algorithms use scheduling heuristics like force-directed scheduling and list scheduling to assign operations to resources in a way that is power-efficient. In order to create the best power-aware schedules, mathematical optimization techniques treat the scheduling problem as an optimization problem and use mathematical programming techniques like integer linear programming (ILP) or mixed integer linear programming (MILP). Furthermore, the review explores power-aware transformations and architectural enhancements that can be integrated with scheduling algorithms to further enhance power efficiency during HLS. These transformations include loop unrolling, resource sharing, and re-timing, among others. Architectural enhancements involve incorporating power-efficient components, such as voltage scaling and power gating, into the generated hardware designs.[3] In addition to discussing the strengths and limitations of existing approaches, the review identifies research gaps and future directions for low power scheduling in HLS. These include the exploration of machine learning-based techniques, leveraging adaptive power management, and considering the impact of other design metrics, such as area and reliability. The purpose of this seminar is to create techniques for high-level design integration of power consumption optimization. The HLS design flow must be modified as a result. A behavioral level description is implemented into a structural description known as a netlist on RTL by the HLS process. The netlist outlines the hardware components—such as ALUs, registers, buses, and wires—that are required to implement the behavior as well as how these components are connected to one another. Additionally, during HLS, a state transition table is created. The netlist's components interact as described in the state transition table, which is a behavioral description of the controller.Thus the HLS consists of three main tasks: scheduling, allocation and binding. The scheduling procedure is expanded in this seminar to find guarded partitions inside the circuit. It is possible to individually activate or deactivate each guarded partition. That implies that such a partition may be deactivated if it is not in use or is idle. Power Scheduler is the name of the developed system. The Power Scheduler technique is depicted in Figure 1 as part of the HLS design pipeline. A Control-Data-Flow-Graph (CDFG) that aids energy minimization is created from the input. The goal is to have a partitioned graph where each partition can be active or deactivated, as was previously indicated.The use of various activation/deactivation methods is permitted by the scheduler. It is possible, for instance, to reduce the power output and prevent the clocking of or to turn off the register's write enable setting.

A. Methodology and Analysis

A digital system is typically designed based on a high-level specification. The specification is converted into an

algorithmic description, similar to source code for C or behavioural VHDL. The behavioral description of the algorithm is compiled into a structural description during the high-level synthesis. Several methods are used in high-level synthesis to carry out the synthesis process. The techniques include scheduling, allocating, and binding. The scheduling procedure arranges the activity in accordance with the timing data. The allocation and binding mechanisms translate the temporally structured operations and memory components to actual resources. In previous approaches, these technologies simply optimized the delay time and the area of the digital system that was being employed. The reduction of electricity consumption is not taken into account in this situation. The reduction of power usage is the primary goal of our strategy. Therefore, the high-level synthesis should incorporate the findings of the activation interval study based on a unique asynchronous architecture. The benefits of bit-serial architectures include the removal of size overhead at the logic level for each operation and the maintenance of parallelism through the use of pipelining design principles. The algorithmic description is converted into a structure internally known as a dataflow graph during high-level synthesis. The operations of the algorithmic description are represented by the nodes in the dataflow graph. Explanation 1: Given a directed dataflow graph G = (V, E) with the set V = v1, . . . , vn of operations within the graph and the set E = e1, . . . ,em for the data dependencies. [1]

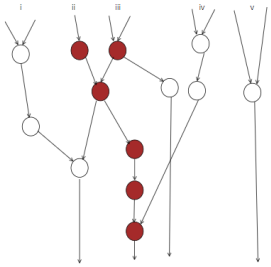

Fig. 1. Dataflow Graph

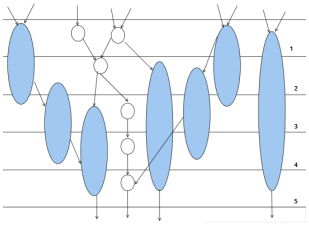

Fig. 2. Dataflow Graph 2

B. Motivation

In fact, it has expanded rapidly, in direct opposition to the chip's entire size. For a chip to be technically and commercially successful today, designing for reduced power has become an essential prerequisite. The difficulty of creating chips with the best energy density and power consumption

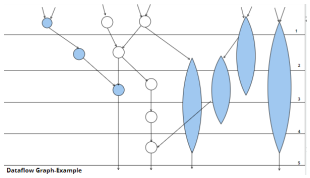

Fig. 3. Dataflow Graph 3

poses a danger to lengthen design cycles and, consequently, time to market. Furthermore, the cost of ownership for both chips and systems may significantly increase if the power challenge is not met. Area and performance have traditionally been the primary driving forces behind ASIC and SoC design, but as more and more designs fail or need to be redone due to power concerns, power reduction has taken the lead. [4] When one asked where the power has be transferred? In this scenario, the power is being classified either Dynamic and Static power dissipation. Logic gates that are changing from one state to another experience dynamic power dissipation. Any internal capacitance connected to the gates transistors must be charged during the switching activity of the gates, requiring power. Additionally, the dormant logic gates are connected to the static power usage. Theoretically, no power should be used by these gates in this situation, but in practice, some leakage current is constantly flowing through the transistors. That means those gates do consume a certain amount of power Usually, the dynamic-power consumption of CMOS logic is usually dominated by the switching of captive loads. Nevertheless, the voltage, the frequency and the switching activity per clock cycle influences the power consumption. [4]

C. Problem Interface:

Low power scheduling for high-level synthesis addresses the challenge of minimizing power consumption in the synthesis of digital systems while meeting performance requirements. It is a critical aspect of the design flow for energy-efficient hardware designs. The problem interface of low power scheduling in high-level synthesis involves the following components: According to [7] High-Level Description: The problem starts with a high-level description of the desired functionality of the digital system. This description can be in the form of a behavioral model or a high-level programming language, capturing the functional requirements of the system. Resource Constraint: The available hardware resources for implementing the digital system are defined. This includes information about the available processing elements, memory components, and other architectural features. The resource constraint sets the bounds on the system's capabilities and determines the potential for power optimization during scheduling. Performance Constraints: The performance requirements of the digital system are specified, which may include factors such as latency, throughput, and timing constraints. These constraints define the desired system behavior and serve as the basis for evaluating the quality of the generated schedules. Power Constraint: The power constraint represents the maximum

allowable power consumption for the digital system. It can be specified as a power budget or as a power limit that the system must not exceed. Minimizing power consumption while meeting this constraint is the primary objective of low power scheduling. Scheduling Algorithms: Various scheduling algorithms are employed to allocate operations from the high-level description onto the available hardware resources. These algorithms aim to optimize power consumption while satisfying the performance constraints. Heuristic-based algorithms and mathematical optimization techniques are commonly used in low power scheduling. The problem interface of low power scheduling for high-level synthesis encompasses the interaction between the high-level description, resource and performance constraints, scheduling algorithms, power-aware transformations, and optimization metrics. By effectively addressing this interface, researchers and practitioners can develop power-efficient hardware designs that meet performance requirements while minimizing energy consumption. [4]

II. High-Level Synthesis Techniques For Low Power Design

High-level Synthesis is described as the process of converting a behavioral description into a structural one. Scheduling, allocation, and binding are three distinct and separate steps in the High-Level Synthesis process. The three phases' relative importance to one another depends on how the design flow develops. Size, performance, and power consumption are the three key criteria that support design decisions. The importance of electricity usage has increased during the past few years. Scheduling operations to reduce switching activity and choosing low power modules while observing timing limitations are the goals of low power high-level synthesis. The works of numerous scholars who have tackled the issue of decreasing power dissipation throughout various stages of High-Level Synthesis are presented in this seminar. [5] Furthermore,These HLS techniques for low power design provide a range of options for designers to reduce power consumption in the synthesis of digital systems. By leveraging power-aware scheduling, transformations, voltage and frequency scaling, power gating, architectural exploration, and power estimation, designers can achieve significant power savings while meeting performance requirements. The choice and combination of these techniques depend on the specific design goals, constraints, and target application domain. [4]

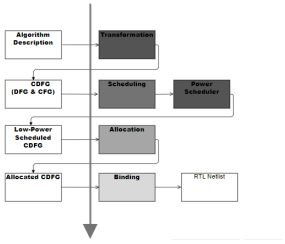

Fig. 4. High-Level Synthesis design flow with Power Scheduler

The power scheduler supports a variety of activation and deactivation methods. It is possible, for instance, to reduce power, prevent register clocking, or set the register write enable to false. Because going from 0 to 1 or 1 to 0 uses energy, the final two approaches avoid altering the register to conserve energy. Achim.R, in his work also states that: The Control-Flow-Graph (CFG) and the Data-Flow-Graph (DFG) are the two graphs that make up the CDFG. Typically, a high-level specification serves as the foundation for digital system design. The specification is converted into algorithmic descriptions, similar to source code for C or behavioral VHDL. The behavioral descriptions are converted into structural ones by the HLS. The internal formats, the CDFG, are changed into during the HLS algorithmic description. The DFG serves as both the data-path for the specified algorithm and the controller for the DFG. The Power Scheduler begins with a CDFG that explains the design. Each node corresponds to operations and control steps, while each directed edge denotes data dependency and control order.

A. Standard Scheduling

From the works of Achim Rettberg,the second step of the developed Power Scheduler is the calculation of the ASAP and ALAP schedules. Both schedules are not really applied to the CDFG. They are only needed to calculate the mobility of each node. ASAP means, all operations are scheduled as-soon-as-possible in the time schedule. Figure 5. shows the ASAP schedule for the example. Respectively, ALAP means to schedule the operations of a DFG as-late-as-possible in time. The ALAP schedule of the design example is shown in figure 6. In both cases we make the following assumptions for the example: • every operations in the DFG have a delay of 1 and • join or fork components need one timestep for their operation. After ASAP and ALAP scheduling it is possible to calculate the so called mobility for each node in the DFG. This mobility measures the degree of freedom for a node to be scheduled. The mobility for a node k of the DFG is defined as

$$mobility(k) = ALAP(k)ASAP(k), \tag{1}$$

whereas ALAP(k) give the timestep where node k is active by ALAP scheduling and ASAP(k) give the timestep where node k is active by ASAP scheduling. Nodes with mobility equal 0 are fixed, because they are scheduled at the same timestep within ASAP and ALAP. For example, the addition operation behind the constant coefficient multiplication cm[0] is scheduled by ALAP at timestep 2 and by ASAP at timestep 1. The mobility of this operation is 1. That is, the node can be scheduled at one timestep. For the developed method it is necessary to calculate for all nodes of the DFG the mobility. This process shows how important the mobility is for the Power Scheduler. The idea is to schedule nodes to same timestep within the DFG and to combine them to partitions. [2]

III. Optimization For Low Power

The created low power approach's major objective is to divide the design during the HLS's scheduling process. Each

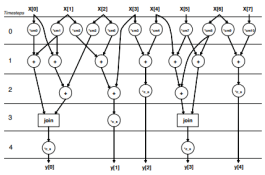

Fig. 5. . ASAP scheduled design example

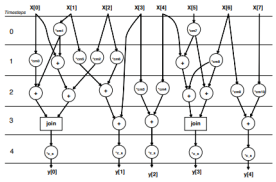

Fig. 6. ALAP scheduled design example

partition enables the design to incorporate specific turn-on and turn-off methods. That implies that a partition can be turned off to save energy if it is not being used. The next sections provide a detailed description of the partitioning. To build a digital system that uses less power, it is essential to optimize power consumption at the highest design level. At various points in the design flow, sacrifices must also be made in order to create the best low power designs, such as timing versus power and area versus power. [2] Engineers that can perform tradeoffs precisely and effectively are essential for successful power-sensitive designs. Engineers must have access to appropriate low power analysis and optimization engines in order to accomplish this, and these engines must be integrated with and used throughout the whole system design cycle. According to [4], there is a defference between dynamic and static power dissipation. Dynamic power dissipation occurs in logic gates that are in the process of switching from one state to another. During the switching activity of the gates, any internal capacitance associated with the gates transistors has to be charged, thereby consuming power. The amount of dynamic power dissipation can be represented by the following equation:

$$Ptotal = Ctotal * V^2dd * fclk, \tag{2}$$

with Ctotal = the total average capacitance being driven/switched, V2dd = the square of the supply voltage, fclk = the amount of activity as a function of the clock frequency.

Additionally, the dormant logic gates are connected to the static power usage. Theoretically, no power should be used by these gates in this situation, but in practice, some leakage current is constantly flowing through the transistors. This indicates that those gates do use some power. The High-Level Synthesis (HLS) phase of the system design flow must be changed in order to incorporate power consumption optimization at the high design level. A provided behavioral level description must be translated into a structural description known as a netlist on Register Transfer Level (RTL) by the HLS process. Additionally, HLS typically handles task binding, allocation, and scheduling. [2]

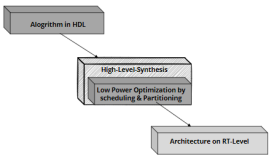

Fig. 7. low power driven HLS

Achim Rettberg also states, that to achieve a developed strategy is to minimize the additional control components needed for the activation and deactivation of the partitions in addition to integrating power saving techniques. Combining partitions could be used to accomplish this. As a result, the goal of our strategy is to consume as little energy as possible. For instants, lets say that our target system consists of n components k1,···,kn. At each timestep a component ki, with i 1,···,n is active or inactive. Then the idea is to build m partitions p1,···, pm whereas each pj, with j 1,···m consists of a set of components ks,···,kr. The idea is to reduce m the number of partitions, because this reduces the additional control overhead for activation and deactivation. This we will achieve by a special scheduling technique.

IV. Operation Implementation

Operation implementation in the context of high-level synthesis refers to the process of mapping and transforming high-level operations from a behavioral description into their corresponding hardware representations. This process involves converting the functional behavior of the operations into a set of hardware components, interconnections, and control signals that realize the desired functionality. Here are the key steps involved in operation implementation during high-level synthesis: Operation Extraction: The first step is to identify and extract individual operations from the high-level description. Operations represent the basic building blocks of the design, performing specific computations or data transformations. Operation Binding: Operation binding involves mapping each extracted operation to a suitable hardware component. This mapping determines which hardware resources will be used to execute the operation. For example, arithmetic operations may be mapped to arithmetic logic units (ALUs), memory operations to memory components, and control operations to control logic. Resource Allocation: Once the operations are bound to hardware components, the next step is to allocate the necessary hardware resources to execute the operations. This includes determining the specific instances of resources, such as ALUs, memories, and registers, required to accommodate the operations' data and control flow. Datapath Design: The datapath design involves organizing and connecting the allocated resources to create the data path that performs the computations required by the operations. The datapath includes the necessary arithmetic and logic units, multiplexers, registers,

and interconnections to implement the desired functionality. The design should consider factors such as data dependencies, timing constraints, and resource availability. Control Unit Design: Alongside the datapath, a control unit is designed to generate the necessary control signals for coordinating the operations and managing the execution flow. The control unit determines the sequencing and timing of operations, including control signals for memory access, data transfer, and control flow decisions. Pipelining and Parallelism: To improve performance and maximize resource utilization, pipelining and parallelism techniques can be applied during operation implementation. Pipelining divides the operations into stages and allows for overlapping execution, while parallelism exploits concurrency by executing multiple operations simultaneously. These techniques can be applied at the operation level or at a higher level of granularity, depending on the design requirements. [7] In high-level synthesis, the operation implementation phase is essential for translating operations' abstract behavior into physical hardware representations. The synthesis process can produce hardware designs that achieve the desired functionality while adhering to performance, space, and power limits by allocating activities to the proper resources, building effective datapaths and control units, and using optimization techniques. [5]

V. Future Directions

Future research in the area of low-power scheduling for high-level synthesis is anticipated to concentrate on resolving current issues and investigating fresh ideas. Here are some probable directions for the future: Deep learning-based algorithms have the potential to be used in low-power scheduling for HLS due to deep learning's growing use across a variety of fields. When scheduling, neural networks can be taught to identify trends and reduce power usage while accounting for complicated connections between processes, resources, and power limitations. In order to identify scheduling policies that are power-efficient, deep reinforcement learning techniques can also be investigated [3]. Using approximate computing techniques, it is possible to trade off computation precision for power efficiency. The goal of future study might be to incorporate approximation computing concepts into the high-level synthesis process. In order to reduce power consumption while maintaining acceptable levels of accuracy, this includes finding computationally complex activities that can tolerate a certain level of inaccuracy and creating algorithms to translate them to approximate hardware implementations. Enhancing the accuracy of power modeling and estimate is essential for successful power optimization during HLS. [6] The development of more precise and fine-grained power models that account for power consumption at many levels of granularity, from individual processes to architectural components, could be the main goal of future research. To improve power estimation accuracy, advanced machine learning techniques might be investigated, such as data-driven modeling and statistical analysis. Memory optimization with consideration for power usage: A large amount of the power used by digital systems is consumed by memory subsystems.

VI. Conclusion

In conclusion, low power scheduling for high-level synthesis is crucial for creating hardware that is designed to be energy-efficient. As high-level descriptions are converted into hardware implementations, power consumption must be optimized while performance standards are met. Designers can dramatically lower power consumption without compromising system performance by utilizing power-aware scheduling algorithms, power-aware transformations, voltage and frequency scaling, power gating, and architectural exploration. Improved energy efficiency, enhanced performance, flexibility, and integration with the high-level synthesis flow are benefits of low power scheduling. Through the use of these strategies, battery-operated devices can have their power consumption minimized, their battery life increased, and digital systems' overall energy usage decreased. Low power scheduling does come with some difficulties and trade-offs, though. Among the drawbacks include increased design complexity, power and performance trade-offs, the necessity for extended design exploration, and taking into account a variety of design metrics and limitations. To achieve the best possible trade-off between power consumption and other design criteria, designers must carefully balance these elements. The study of deep learning-based techniques, approximate computing, improved power modeling and estimation, power-aware memory optimization, heterogeneous and reconfigurable architectures, multi-objective optimization, power-aware design space exploration, and energy-aware design are some future directions in the field of low power scheduling for high-level synthesis. Researchers and practitioners may improve the power efficiency of digital systems, support emerging technologies, and facilitate the creation of energy-efficient and sustainable hardware designs for a variety of application domains by addressing these future directions. Future energy-efficient and high-performance digital systems will continue to be in great demand, and low power scheduling approaches will be essential to supplying that demand.

References

[1] Achim Rettberg - Low Power Driven High-Level Synthesis for Dedicated Architectures.

[2] *A Fast Asynchronous Re-configurable Architecture for Multimedia Applications. In Proceedings of the 14th Symposium on Integrated Circuits and System Design (SBCCI 2001), Pirenópolis, GO (Brazil)*, 2001.

[3] Anand Raghunathan, Niraj K. Jha, and Sujit Dey. High-Level Synthesis for Low Power. In Anand Raghunathan, Niraj K. Jha, and Sujit Dey, editors, *High-Level Power Analysis and Optimization*, pages 115–153. Springer US, Boston, MA, 1998.

[4] Achim Rettberg. *Low Power Driven High-Level Synthesis for Dedicated Architectures*, volume 22 of *C-LAB Publication*. Shaker Verlag, Aachen, 2007.

[5] Achim Rettberg, Bernd Kleinjohann, and Franz J. Rammig. Integration of Low Power Analysis into High-Level Synthesis. In Bernd Kleinjohann, K. H. Kim, Lisa Kleinjohann, and Achim Rettberg, editors, *Design and Analysis of Distributed Embedded Systems*, volume 91, pages 195–204. Springer US, Boston, MA, 2002. Series Title: IFIP Advances in Information and Communication Technology.

[6] Achim Rettberg, Bernd Kleinjohann, and Franz J. Rammig. Integration of Low Power Analysis into High-Level Synthesis. In Bernd Kleinjohann, K. H. Kim, Lisa Kleinjohann, and Achim Rettberg, editors, *Design and Analysis of Distributed Embedded Systems*, volume 91, pages 195–204. Springer US, Boston, MA, 2002. Series Title: IFIP Advances in Information and Communication Technology.

[7] Zhiru Zhang, Deming Chen, Steve Dai, and Keith Campbell. High-level Synthesis for Low-power Design. *IPSJ Transactions on System LSI Design Methodology*, 8(0):12–25, 2015.

DECLARATION OF ORIGINALITY

I, Ukamaka Akumili Anaedu , herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

Ukamaka Akumili Anaedu Date