Low Power Scheduling For High Level Synthesis

*This is the seminar paper for hardware-software co-design

1st Aditya Kumar
dept. of Electronic Engineering
Hochschule Hamm Lippstadt
Lippstadt, Germany
aditya.kumar@stud.hshl.de

***Abstract*—High-level synthesis is the translation process from behavioural definition to structural definition. In the recent days it has become increasingly important to be power efficient in these processes, extending high-level synthesis to cover power optimisation. There are many methods developed to achieve this. One such method integrates low power consumption in the scheduling process of the High-level Synthesis. The following explains and summarises this methodology.**

I. Introduction

Power management in application specific integrated circuit (ASIC) and systems-on-chip (SoC) has become very important specially when one considers the increased use of low-power, battery-operated microelectronics. These devices are becoming physically smaller and are expected to have increased functionality. It has become critical for a chip to consume less power to be a commercial success.[1]

While we try to shrink these chips, static power is one such element that is growing inversely proportional to the overall size of the chip. Therefore it has become critical for a design engineer to apply a reliable power network and minimise power dissipation. One can distinguish between static and dynamic power dissipation. Static power consumption is associated with the logic gates when they are not active. Theoretically, the gates should not consume any power but there is always some amount of leakage current passing through the transistors stating that the gates so consume a certain amount of power.

Where as, Dynamic power dissipation occurs in logic gates that are in the process of switching from one state to the other. Any internal capacitance associated with the gates transistors has to be charged during the switching activity of the gates. The following equation represents the amount of dynamic power dissipation in the logic gates :

$$P_{total} = C_{total} * V_{dd}^2 * fclk, \tag{1}$$

with C_{total} = The total average capacitance being driven/switched, V_{dd}^2 = The square of the supply voltage, $fclk$ = The amount of activity as a function of the clock frequency.

Thus it has become important to alter the process of system design flow called High-Level Synthesis(HLS). In this paper we will be discussing the implementation of low power methods within the scheduling process of HLS using a power scheduler described in [2].

II. Methodology

To integrate low power methods within the scheduling process of the High-level Synthesis, the power scheduler generates a scheduled Control-Data-Flow-Graph (CDFG) from an unscheduled CDFG, that supports power saving.

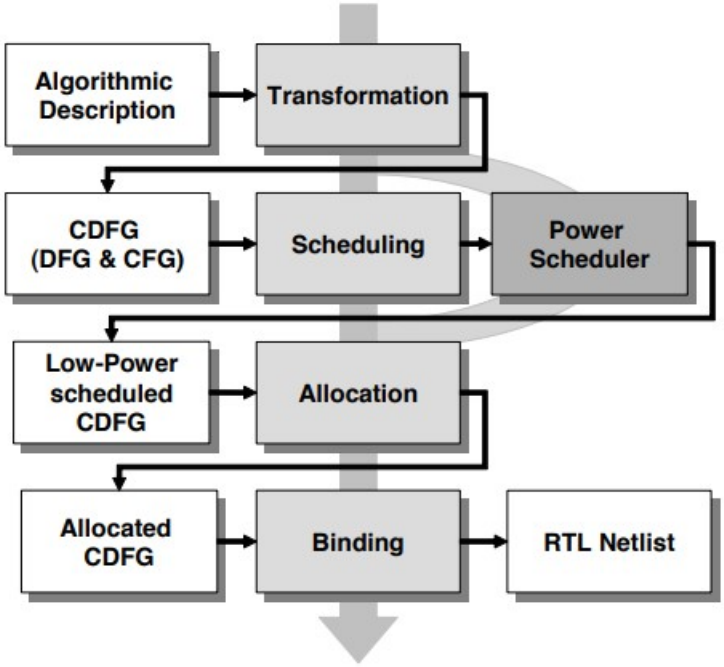

Fig. 1. HLS embedded with a power Scheduler

It uses a partitioned graph, where a guard could activate or deactivate each partition. Gated clocks, Guarded Evaluation or power down could be used to implement this guard. The following subsection elaborates the different scheduling phases of a power scheduler.

Scheduling Flow

The CDFG incorporates two inter-weaved graphs, a so called Control-Flow-Graph (CFG) and a Data-Flow Graph (DFG). They are defined as follows:

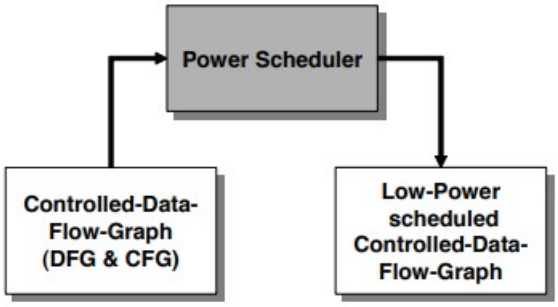

Fig. 2. Power Scheduler

definition 2.1 (Control-Flow-Graph(CFG)): Let $G = (V_c, E_c)$ be a directed graph, with the set of nodes V_c = $v_1, v_2, , v_n$. Each v_i is a control object of a given algorithm. An edge (a_s, a_t) with $a_s, a_t \in$V$_c$ describes the transition from one control object to another one. This can also be interpreted as the transition from one state of the system to the following stat.

definition 2.2: Let $G = (V_d, E_d)$ be a directed graph, with the set of nodes V_d = $v_1, v_2, , v_n$. Each v_i corresponds to an operation, a fork or a join of an algorithm. An edge (a_s, a_t) with $a_s, a_t \in$V$_d$ describes the data dependencies between the different nodes.

Usually, the design process of a digital system is based on high-level specification being transformed into an algorithmic descriptions, such as c or behavioral VHDL source code. During the High-level Synthesis the behavioral description is transformed into the structural description. During the HLS algorithmic description are transformed into internal formats. The CDFG acts as the controller for the DFG which itself is the data-path of the given algorithm. The Power Scheduler begins with a CDFG, that describes the design, where each node corresponds to operations and control steps, and each directed edge represents data dependency and control order.

1) The first step the Power Scheduler reads the CDFG that consists of a CFG and DFG, and stores these graphs in the internal data format.
2) In the second step, the Power Scheduler performs As-Soon-As-Possible (ASAP) and As Late-As-Possible (ALAP) scheduling algorithms on the DFG. see figure 3. In ASAP scheduling all operations are scheduled as soon as they can be processed in the given sense of time. On the other hand, in ALAP scheduling all operations are scheduled as late as possible. Although these schedules are not really applied to CDFG, these scheduling methods are processed to calculate the mobility of each operation within the DFG. Mobility stands for the degree of freedom an operation has within the scheduling.
3) The next step in this flow is an important one. In the third step, the Power Scheduler calculates the active/inactive paths during operation within the DFG. In this process all backward and forward edges of the paths are analysed. This analysis consists of three different phases.
   - In the first phase, it examines all disjoint paths of the DFG. In due course some of these paths are not active during the entire run-time of the system.
   - In the second phase, we examine the fork and join nodes of the DFG. The partition is constructed on the bases of different paths between the fork and join nodes, as they are alternatively active during the run-time.
   - In the third phase, we examine the control nodes of the CFG. That means, that different paths that are alternatively active during the run-time can be identified if it is relevant to schedule them as soon as possible. These examined paths are the basis of partitioning process and could be combined to partitions again. All paths are nodes in a so called compatibility graph.
4) In the fourth step, the Power Scheduler combines the paths that are calculated in the second step to build the partitions, see fig 3. To achieve this, it is necessary to examine if conflicts between the paths exists. Two paths are in conflict to each other if both of them are depending from the same fork, join or control node. Each of these paths corresponds to a node in the compatibility graph. The edges in this graph show if the nodes are compatible or not. Further, the compatible nodes can be combined to a partition which then can either be turned on or off. This means that if the paths are in conflict there will be no edge in between them in the compatibility graph. Then a clique search algorithm is used to find cliques in the compatibility graph. Finally, a clique builds a partition than can be activated or deactivated during the run-time to save energy.

The Power Schedulers scheduling has optimization goals as well . The mobility of each node is computed using the ASAP and ALAP scheduling methods. After all pathways are known, the mobility of the paths can be computed since paths are made up of nodes, and the mobility of nodes is known. The compatibility of the nodes (=paths) in the compatibility graph refers to data dependencies in terms of time and mobility.

Inserting activation or deactivation methods into a circuit incurs power costs in both the data stream and the controller. To eliminate these extra expenses, the routes must be combined into partitions. Nonetheless, it is feasible that a partition has just one path following the clique method.

III. Scheduling Under Low power Constraints

The following is an example of the scheduling under low power constrains and is stated in[3]
The goal is to find a schedule that reduces the Power consumption defined in the following formula :

$$Power_s = \sum_{i=1}^{n} p_i * m_{is} * D_{is} + g_s. \quad (2)$$

- m_{is} is the number of real resources of a certain type in a schedule s.

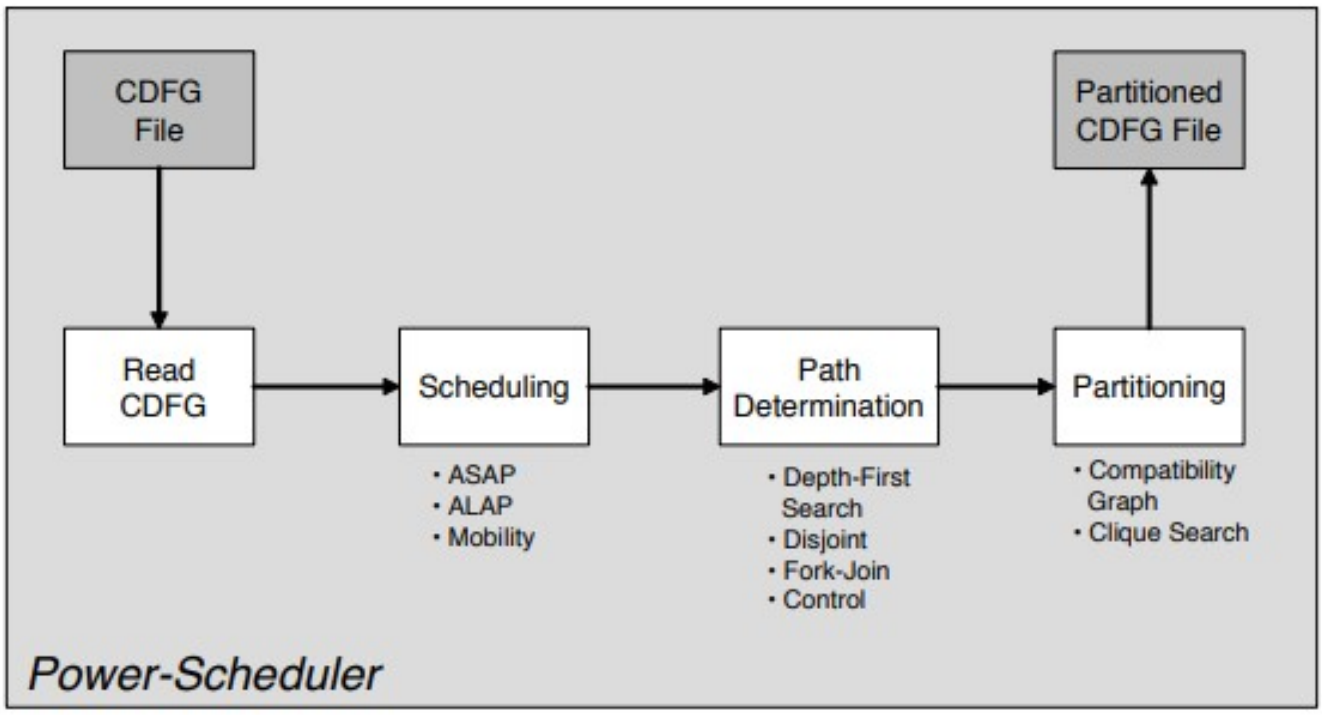

Fig. 3. Flow of Power Scheduler

- p_i is the power consumption of real resource i.
- D_{is} is the duration of node i within a schedule.
- g_s is the additional cost for switch-on/off mechanism for schedule s.

In [4] The activation interval analysis calculates three different partitions for the DFG in figure 4 that has 10 operations. Each of these partitions can be turned on/off individually. From left to right the first partition contains A. The one in the center contains B,C,E,H and J. The one on the right contains of D,F and G. One can tell that its not necessary to activate Node C in Partition for the entire run time.

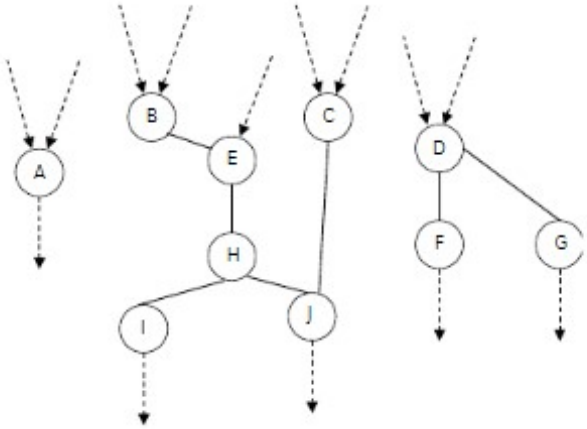

Fig. 4. Data Flow Graph

Mobility described in the scheduling flow is the calculated with the ASAP and ALAP scheduling[5]. It is possible to compute these schedules in linear time. The mobility is calculated by the execution time interval of the two schedule for a Node V of the DFG

$$Mobility_v = alap_v - asap_v + 1$$

The execution timing intervals for this design is represented by figure 7.

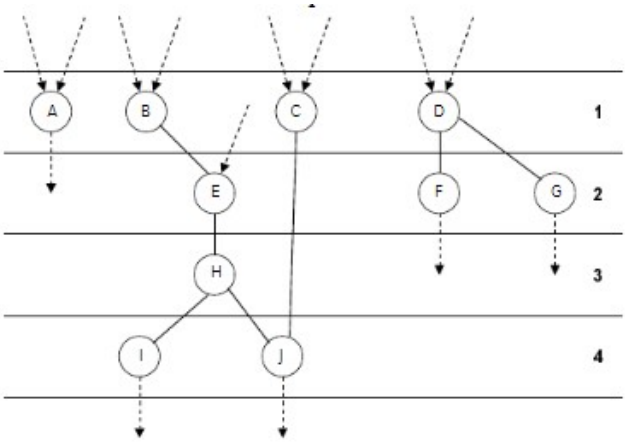

Fig. 5. DFG with ASAP schedule

The Mobility for each node will be :

- $Mobility_A = 4$
- $Mobility_B = Mobility_E = Mobility_H = Mobility_I = Mobility_J = 1$
- $Mobility_c = Mobility_D = Mobility_G = Mobility_F = 3$

The operations can be executed in exactly one time frame if there mobility is equal to 1 and different time frames if the mobility is greater than one. This means that the operations that are relevant for power optimization can be be accessed by the ON/OFF mechanism.

The combination of these operations can maintain a guarded partition, but the latency, power consumption, area for each resource, and data dependencies must all be considered. Here real resources are not saved, but that activation and deactivation mechanisms such as gated clocks and guarded evaluation[2][6] that are explained in the next section are integrated.

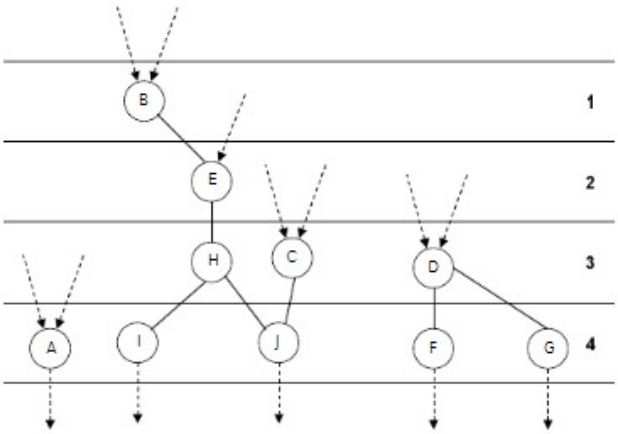

Fig. 6. DFG with ALAP schedule

Keeping the data dependencies and the mobility explained above, Operation A can be said to be completely independent from all the other operations and can be executed at any of the given cycles. In addition the operations can form an individual partition that can be activated or deactivated. Non the less, the goal here is to combine them to a guarded partition that are in the same time frame.

As stated before D, F and G can be executed within 3 cycles whereas operations B, E, H, I, and J have mobility 1 and will not be considered in this analysis, but those actions establish another partition. At the given point the method described yielded the same results as the activation interval analysis in [4]. Operation C is not assigned to a partition. The mobility of operation C is three and must be executed before the computation of operation J can begin. This must be done within the first three cycles. As a result, operation C can be turned off for two cycles. Therefore contributing in low power consumption.

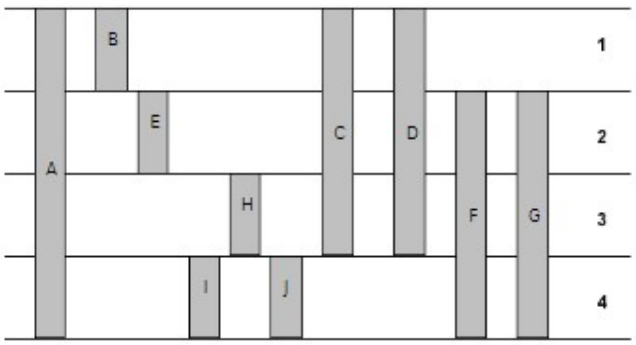

Fig. 7. Execution Intervals of the Operations

Figure 8 depicts one possible split of the entire data-flow graph. With gated clocks or guarded evaluation, each partition can be activated or deactivated. If a gated clock is utilised, the additional costs are based on four [AND] gates (one for each partition).

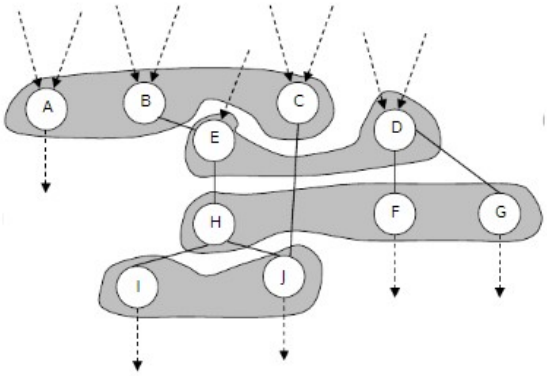

Fig. 8. Partitioned DFG

To summarise, this method is based on recognising the data dependencies of operations from the data-flow graph, after which, the ASAP and ALAP schedules are generated to determine the execution intervals and mobility. The activation and deactivation of partition is determined in conjunction with the mobility and data dependencies. The preliminary findings of various tiny filter methods for compression demonstrate that it is possible to achieve up to a 20 % reduction in power usage. When compared to ordinary bit-serial implementation, adopting low power high-level synthesis for a bit-serial design can save up to 10% of the power consumption for the typical high-level synthesis benchmark.

IV. POWER REDUCTION METHODS

Following are the power reduction methods used within the power scheduler. One can either use one of these or combine all of them in the design.[2]

A. Gated Clock

Gated clocks shown in figure 9 can be used to reduce dynamic power. the clock signal is received by an AND gate.for example, the clock signal is directed to the storage elements of the logic gate only the clock enable signal is true. Therefore, only the storage elements are driven by a clock. Here the dynamic power consumption is reduced when the clock is not directed to storage elements, setting the switching activity to zero.

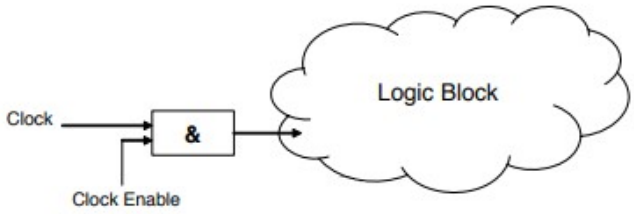

Fig. 9. Example of a gated clock implementation

B. Guarded Evaluation

This is similar to the gated clock. here a Guard is introduced that corresponds to a storage element with a write enable signal. if this signal is set to false the register(storage element) wont write the incoming data to the output, thus creating a switching activity behind the block therefore reducing power consumption.

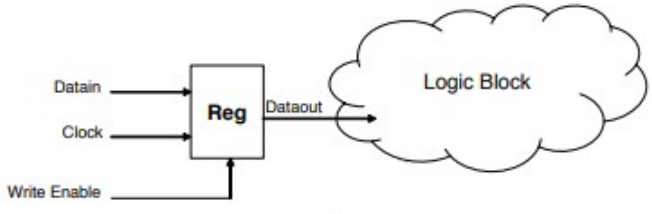

Fig. 10. Guarded evaluation implementation

C. Power Down

This method, in comparison to others can be used to reduce both dynamic as well as static power consumption. Figure 11 shows a top level view of a chip with three areas whereas each of them has own power pins. These pins are controlled separately which means that if the power goes down in one area there is no switching activity which would lead to reduction of dynamic power consumption. On the other hand, everything will we switched off thus reducing static power consumption.

V. Conclusion

In this paper, a low power driven approach for high level synthesis using a power scheduler is discussed. the scheduler reads the CDFG to writes scheduled and partitioned CDFG. Partitions which can be turned on/off are based on the path determination. An example is given on how the partition of DFG can be analysed and and methods for power reduction are discussed.

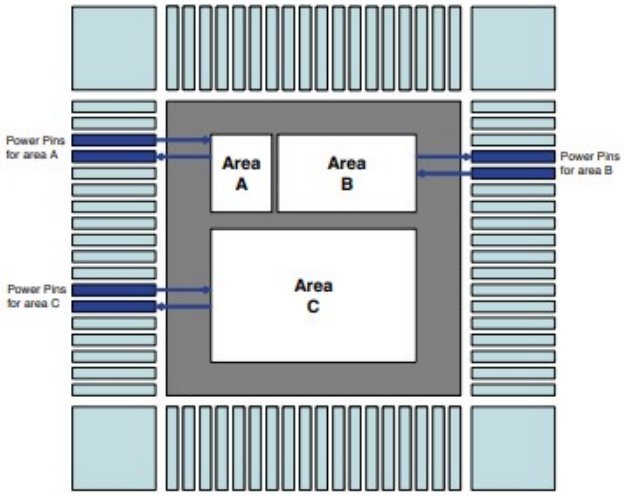

Fig. 11. Split chip area to implement power down

Acknowledgment

I herewith declare that I wrote this paper on my own and did not use any unnamed sources or aid. Thus, to the best of my knowledge and belief, this paper contains no material previously published or written by another person except where due reference is made by correct citation.

References

[1] Magma-Design-Automation. Enabling Low Power Design Within an RTL-to GDSII Implementation Flow. White Paper, 20.

[2] A. Rettberg "Low power Driven High-Level Synthesis for Dedicated Architectures" chapter 3, page 37, 2006

[3] Rettberg, Achim and Kleinjohann, Bernd and Rammig, Franz, "Integration of Low Power Analysis into High-Level Synthesis" 2003.

[4] A. Rettberg, B. Kleinjohann, W.Hardt " Using Activation Interval Analysis for low power". In Proc. of the 9th NASA Symposium on VLSI Design,Albuquerque, New Mexico, November, 2000

[5] H. Krämer, W.; W. Rosenstiel, "System Synthesis Using Behavioral Description", European Design Automation Conference (EDAC), pages 227-282, 1990

[6] L. Benini, G. De Micheli, "Transformation and Synthesis of FSMs for low-power gated-clock implementation". IEEE Transactions on Computer-AidedDesign , 1996