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.the following paper will be discussing various methods created**

***Index Terms*—component, formatting, style, styling, insert**

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.

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.

II. Power Scheduler

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.

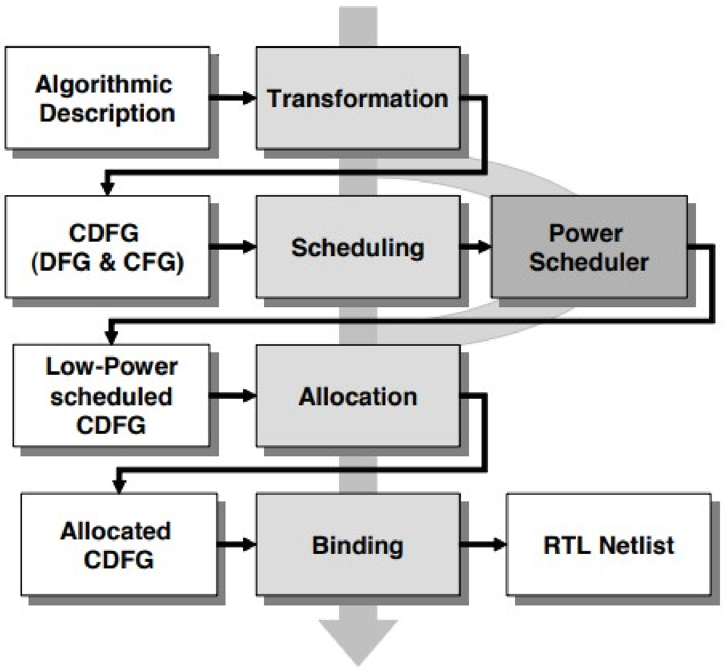

Fig. 1. HLS embedded with a power Scheduler

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.

A. 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:

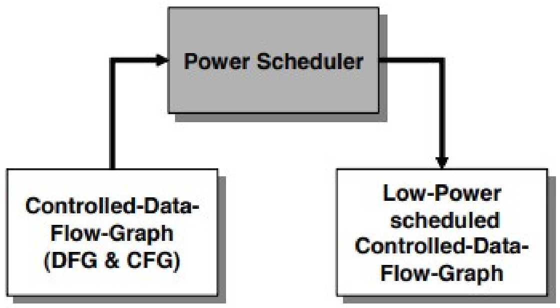
Power Scheduler
Controlled-Data-Flow-Graph (DFG & CFG)
Low-Power scheduled Controlled-Data-Flow-Graph

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 there are so-called conflicts between the paths. 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.

From the perspective of the reader of this paper, two questions are opened. Why is it important to build partitions to save energy instead of controlling each single node? Why not use each calculated path as a partition? The answer to both questions is the same. The insertion of activation or deactivation mechanisms into a circuit has also power costs in the data-path as well as in the controller. To reduce these additional costs it is necessary to combine the paths to partitions. Nevertheless, it could be possible that after the clique approach a partition contains only one path.

III. 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.

A. Gated Clock

Gated clocks shown in fig 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

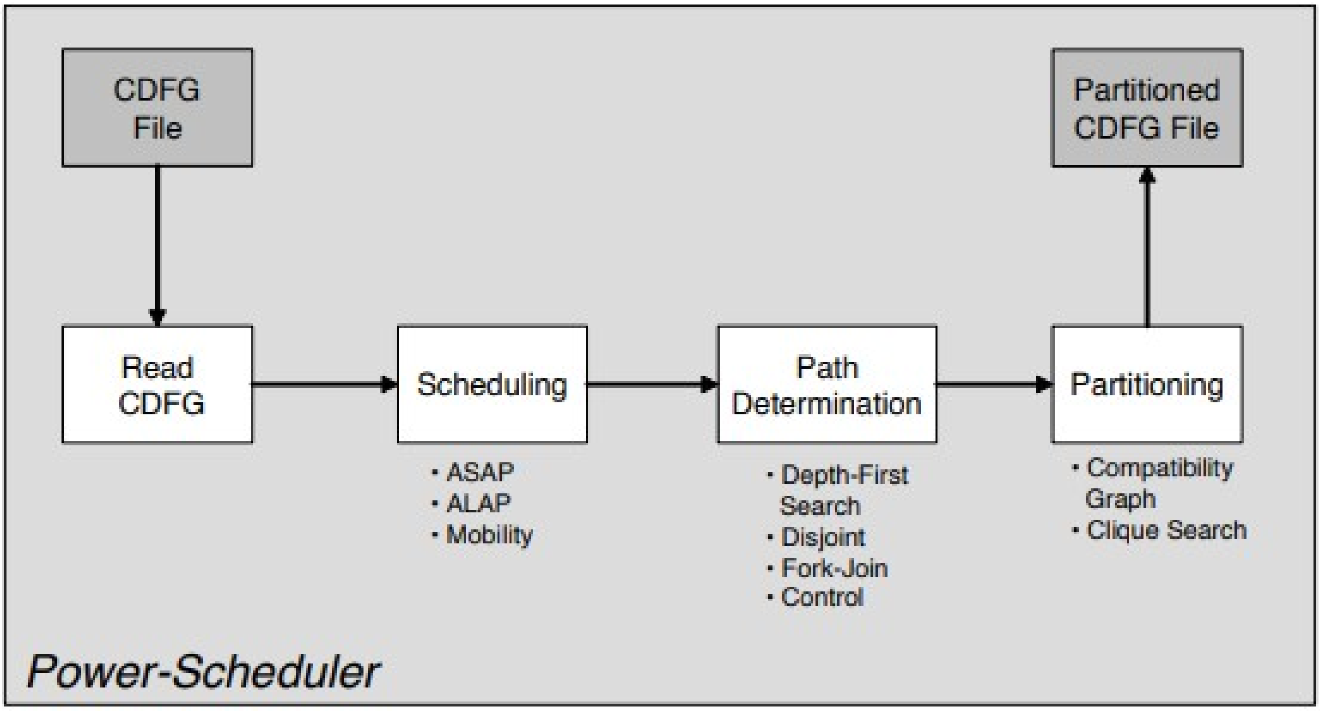

Fig. 3. Flow of Power Scheduler

directed to storage elements, setting the switching activity to zero.

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.

C. *Power Down*

This method, in comparison to others can be used to reduce both dynamic as well as static power consumption. Figure 3.6 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.

ACKNOWLEDGMENT

REFERENCES

REFERENCES