Low Power Scheduling for High-Level Synthesis

Muhammad Farid Izwan Bin Mohamad Shabri
Hochschule Hamm-Lippstadt
B.Eng. Electronic Engineering
Lippstadt, Germany
muhammad-farid-izwan.bin-mohamad-shabri@stud.hshl.de

***Abstract*—This paper explains how low power scheduling is used for High-Level Synthesis . Scheduling is one of many stages for high-level synthesis. There many parameters that are need to be considered such as switching activity to reduce or lower the power consumption which also meet the time constraints of the tasks that are being scheduled. In this paper, the strategies to have low power, hence optimise the power during scheduling process are discussed more details, so that we clearly understand the benefits of low power scheduling for High-Level Synthesis.**

I. Introduction

In this world full of advanced technology, microelectronics is very vital and act as a backbone to the development of software and hardware systems in recent decades. The real embedded system, integrated circuit IC and robot automation are example of hot topics that many developers and engineers discuss on how to improve the creation of new advanced technology. Design, Fabrication, Testing and Packaging are four stages in the creation of integrated circuit. Hence, scheduling process is very vital to make sure the system run smoothly with all the time requirement for all hard or soft real time task can be met [1].

In recent years, the usage of power consumption is very big. This because of high demand for IC and SoC devices, so it drives to consume the smallest amount of power strategy. Furthermore, creating optimal low power designs is not easy but surely possible. But there are many factors that must be considered, since there is also a demand in many electronic devices especially telephone to have longer battery life and increase the functionality. In the case of today extremely large and complex designs, implementing a reliable power network and minimizing power dissipation have become major challenges for design engineers [3].

This paper is organized as follows. Section II describes briefly high level synthesis which have one of the important process called scheduling process. Next, Section III revises topic power consumption and dissipation. Section IV explains shortly about scheduling algorithm. Section IV explains more details scheduling process under low power driven. There many subsections will be discussed such as, methodology, example, result, etc.

II. High Level Synthesis

In this section, High Level Synthesis will be explained more details. This is to get the surface of idea before going deeper into low power scheduling.

A. Definition

High-level Synthesis is a translation process from behavioral description into a structural description. Scheduling, allocation and binding are three phases of High-Level Synthesis. The phases are randomly ordered based on design flow and are used to support decision making on design stage , namely size, performance and for this topic, we focus on power consumption that has become more and more important in recent years. When talking about power consumption, it means that we wanted to optimize the usage of energy for a system so that it become more efficient. The main purpose of low power scheduling in High-Level Synthesis is to schedule operations while minimizing switching activity and select low power modules while meeting the time constraints.

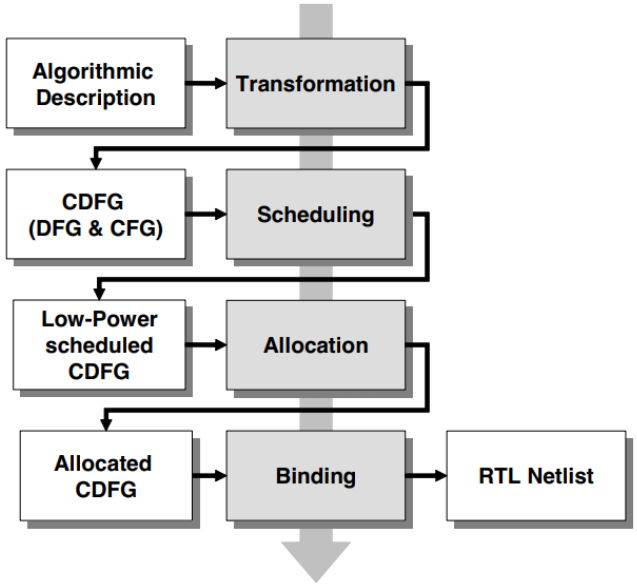

Fig. 1. Design phases in High Level Synthesis [1]

B. Fundamentals of High-Level Synthesis

As mentioned, the three phases are scheduling, allocation and binding. In scheduling phase, the time for operation to be executed is determined. The allocation phase determines

how many instances of each resources are required and the binding or also well-known as assignment phase selects on what resource a computational operation will be run. “Fig. 1” shows design phases of the High-Level Synthesis. Most of the time, the input for High-Level Synthesis is a control data flow graph, called CDFG.

III. Power Consumption and Dissipation

In recent years, the demand of personal computing devices and wireless communications equipment has increased. As a result, there is a greater demand for low-power circuit design. Along with area and run-time, lowering power consumption has become one of the most essential criteria. There are some reasons why the consideration of low power is very strong [1] :

- Increased demand of portable systems which require the battery life to be more long lasting. Laptops and mobile phones are the big picture for this issue.
- Thermal considerations which means that the cost of cooling and packaging would be reduced if low power dissipation.
- Environmental issues because when smaller power is dissipated, lower the heat pumped into the rooms which means that usage of electricity is reduced, hence give less impact on environment.

The source of power consumption or dissipation in digital CMOS circuit are as shown in “Fig. 2”.

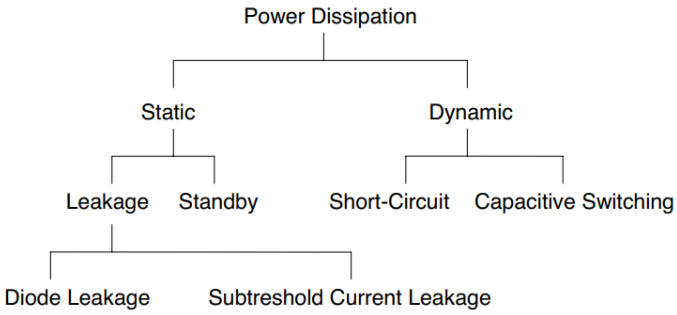

Fig. 2. Source of Power Dissipation [4].

The parameters that are independent and not related to each other to affect power consumption as well as energy usage are [4]:

- supply voltage
- the clock frequency
- the switching activity per clock cycle at various signals in the circuit.
- the parasitic capacitance

The parameters above are proven by an expression, called dynamic power dissipation expression:

$$P_{dynamic} = \frac{1}{2} * C_L * V_{dd}^2 * a * f \quad (1)$$

C_L is the load capacitor, V_{dd} is the supply voltage, a is the average or expected number of transitions per clock cycle also known as the switching activity, and f is the clock frequency.

Hence, to optimize the power consumption, the factors must be tackled. Here are the suggestions or ideas:

- using the smallest possible supply voltage.
- parallelism and pipe-lining mechanism to lower required frequency operation.
- power control by disconnecting the power source when the system is in idle mode.

IV. Scheduling Algorithms

Scheduling is one of the vital phases in High-Level Synthesis. As depicted in “Fig. 3”, there are different types of scheduling algorithms. As mentioned before, there are few main factors in scheduling process that give impact on reducing power consumption which are supply voltage, clock frequency, switching activity mechanism etc. In this section, scheduling algorithms for these mechanisms will be explained clearly on how these factors can be considered to lower power consumption.

A. Scheduling Algorithm with Multiple Supply Voltages

Multiple supply voltage design may affect the power consumption. Since the voltage used can be minimized, hence power consumption will be reduced as a result. For example, Multiple Voltage Scheduling (MVS) algorithm method [3].

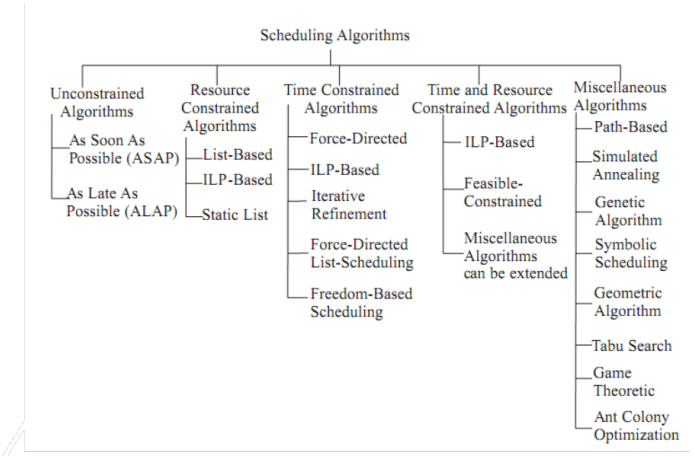

Fig. 3. Types of scheduling algorithms [1]

B. Scheduling Algorithm based on Frequency Variation

As discussed earlier, by reducing of clock frequency can save more power consumption. For example, Dynamic Frequency Clocking (DFC) is introduced and has proven this method. DFC generally have 3 concepts which are reconfigurable architecture, frequency synthesizer and clock dividing strategy.

C. Scheduling Algorithm based on Switching Activity

Last but not least, the factor that is can be considered and can effect the power consumption is switching activity event. Profile Driven Synthesis System (PDSS) is introduced and also known as the synthesis flow of the synthesis system. PDSS operates a constraint-satisfying design with the least amount estimated switching activity. This could be more helpful in reducing the power consumption.

In the next section, scheduling process under low power consumption is explained more details.

V. Low Power Scheduling

The focus in this topic is how power consumption is optimized therefore low power is dissipated in scheduling task in High-Level Synthesis. The growing use of battery-powered and often wireless portable systems is boosting demand for low-power IC and SoC technologies. Furthermore, users have come to expect more capability and longer battery life from such systems. A modern cellular phone, for example, might have capabilities like the ability to operate as a personal digital assistant, storing data such as photos and connecting to the Internet. Implementing a stable power network and limiting power dissipation have become key issues for design engineers in today's exceedingly big and complicated architectures.

A. Methodology analysis

To integrate low power methods in High-Level Synthesis, a system called Power Scheduler is developed as described in "Fig. 4" [1].

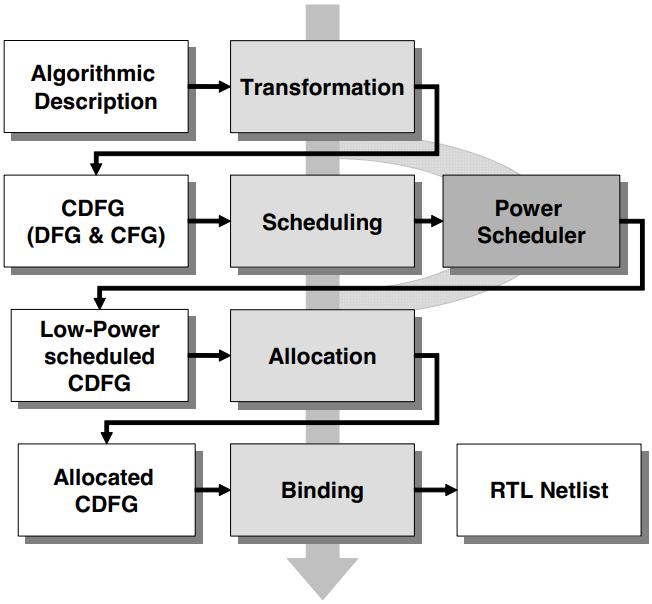

Fig. 4. Power Scheduler embedded into High-Level Synthesis [1]

As shown in "Fig. 4", we can see the effect or impact of Power Scheduler in scheduling process. It replaces the existing scheduler of High-Level Synthesis system which causes low power consumption at scheduling process before entering allocation phase.The scheduled Control Data Flow Graph is generated which support low power. To generate the graph by using partition method, so that each partition can be activated or deactivated, while combining partitions to be run, hence minimizing the additional control components which causes less energy usage.

Control Data Flow Graph (CDFG) consists of two interweaved graphs, Control-Flow Graph (CFG) and Data-Flow Graph (DFG). Digital systems are typically designed using a high-level specification. The specification is converted into algorithmic descriptions, such as source code in C or behavioral VHDL. The HLS converts behavioral descriptions to structural descriptions. The CDFG is translated into internal formats during the HLS algorithmic description. The CDFG represents the controller for the DFG, and the DFG represents the datapath of the specified algorithm. Each node corresponds to actions, and each directed edge represents data reliance.

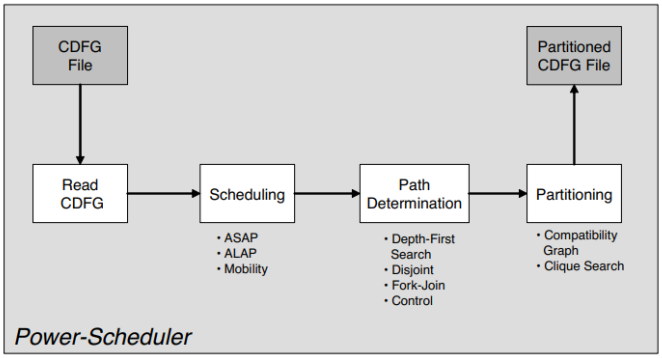

Fig. 5. Steps of Power Scheduler [1]

The "Fig. 5" above describes the steps of the Power Schedular. Here are the more detailed explanations below :

1st Step: CDFG file are read and store the graph into internal format.

2nd Step: Use As-Soon-As-Possible (ASAP) and As-Late-As-Possible (ALAP) scheduling on Data-Flow Graph (DFG). When both scheduling approaches are used, the mobility of each operation inside the DFG may be calculated. The degree of freedom that an operation has within the constraints of its schedule is referred to as mobility.

3rd Step: Calculating the path. Path determination means that during the operation, active or inactive paths are recognized by system. As shown in figure above, there are different layers where the path is analysed. The first layer or level is examination or checking all disjoint paths of Data-Flow Graph. Eventually, some of these pathways are inactive throughout the duration of the system's operation. The next phase is fork and join nodes. Because they are alternately active throughout run-time, the distinct pathways between the fork and join nodes serve as the foundation for the partitioning design.The third phase is control nodes. That is, if it is appropriate to schedule them as soon as feasible, multiple pathways that are alternately active during run-time can be found. The evaluated paths serve as the foundation for partitioning and are merged

to form partitions. All pathways are nodes in a graph called compatibility graph.

4th Step: to construct the partitions by merging the pathways obtained in the second step. Before doing this, checking if there are conflicts between the paths is a must. Two pathways are in conflict if they both originate from the same fork, join, or control node, for example. Each path corresponds to a node in a graph of compatibility. The edges of the compatibility graph indicate whether or not the nodes are compatible with one another. That is, compatible nodes can be combined to form a partition that can be activated and deactivated. A full discussion of the generation of the compatibility graph, particularly for the edges, will follow. There is a description of a clique search algorithm that is used to discover cliques in a compatibility graph. Finally, a clique constructs a partition that can be enabled or deactivated during run-time to conserve energy. The CDFG is involved in this step.

B. Power reduction method

There are some methods that are recommended to minimize power dissipation during scheduling. The methods are listed below with short explanation.

1) Gated Clock: By implementing of a gated clock as shown in "Fig. 6", the dynamic power is decreased. An AND gate receives the clock signal. If the clock enable signal is set to true, the clock is routed to the logic block's storage components. As a result, only storage elements and no arithmetic logic are driven by a clock signal. A clock is required by an ALU (arithmetic logical unit) for an internal carry register. However, if the clock is not directed to the storage elements, the switching activity is reduced to zero, which reduces dynamic but not static power usage.

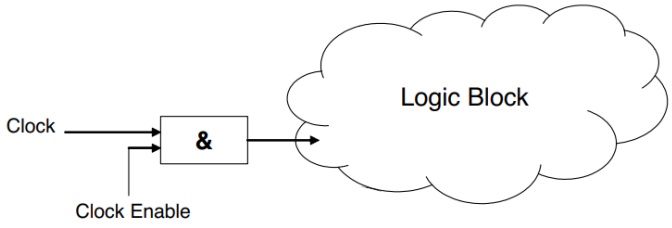

Fig. 6. Example implementaion of Gated Clock [1]

2) Guarded Evaluation: In this scenario, a guard corresponds to a storage element (such as registers) with a write enable signal, as shown in "Fig. 7". The registers do not write incoming data to the output if the write enable signal is set to false. As a result, there is now switching activity in the logic block below the register, which reduces dynamic power consumption in the same way that the gated clock approach does.

3) Power Down: It is feasible to lower both dynamic and static power usage by using power down. "Fig. 8" shows a top-level view of a chip with three sections, each with its own set of power pins. These power pins can be controlled independently. That is, if the power goes out in one location,

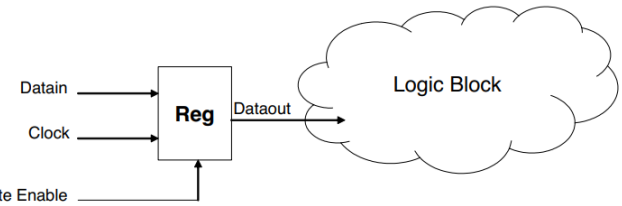

Fig. 7. Example implementaion of Guarded Evaluation [1]

no switching activity occurs. This reduces dynamic power usage, but it also reduces static power because everything is turned off. In the next part, path determination which one of important strategies in having lower power consumption will be discussed more details.

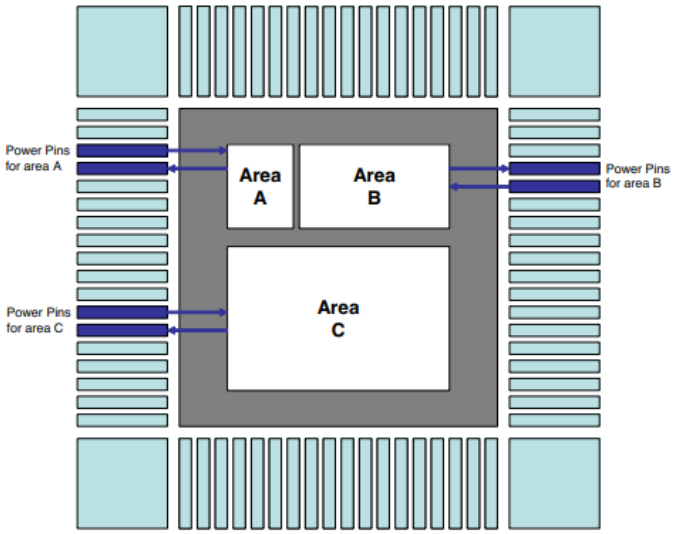

Fig. 8. Example of power down implementation y splitting chip areas [1]

C. Path Determination

As discussed earlier, path analysis is one of the important aspects to design low power scheduling. For path determination, there are 3 main aspects need to be examined:

1) Independent paths or disjoint paths: If an operation is on independent path, it could be waste of power if the operation is activated for the entire of running time. To reduce the power, the independent path just need to activate when it is required and deactivate when it does not operate.

2) Paths between fork and join nodes: In the most cases, paths between fork and join elements in a system design are realized by multiplxer and demultiplexer. The idea behind this path is to make only one active path while the rest is not active.

3) Paths influenced by Control Nodes: By lowering the amount of unused computation ,it could help in reducing power. That means the nodes can be controlled by certain condition to be activated. In normal design, data flow is

determined at run time based on conditions derived from the input. It's just a waste of power if two events running but producing the same results. Hence, by setting the condition, only one of the event could be executed, as a result.

D. Clique Problem

Clique search problem will be described in this section. Clique consists of combination of different paths that have been divided into partition. To reduce the power usage, this partition can be activated and deactivated. Later, the partition is integrated into clique algorithm. Hence, the nodes that are in timing and dataflow conflict to the nodes in the clique can be detected to remove the problem.

Besides, there are two different clique search strategies. First strategy is to find all cliques of a graph and another strategy is to find only maximum clique. Later, delete the nodes and this steps will be repeated until there is no clique.

Algorithm 1 Clique algorithm

```
0: proc max_clique(1, ..., v_n}, n, depth, m)
0: CBC = depth
0: for i = 1 to n do
0:   if (depth + (n − i)) > CBC then
0:     if depth = 1 then
0:       m = m + 1;
0:       C_m = ∅;
0:     end if
0:     C_m = C_m ∪ {v_i};
0:     k = 0;
0:     L_vi = ∅;
0:     for j = i + 1 to n do
0:       if v_i adjacent to v_j then
0:         L_vi = L_vi ∪ {v_i};
0:         k = k + 1;
0:       end if
0:     end for
0:     if L_vi ≠ ∅ then
0:       max_clique(L_vi, k, depth + 1, m)
0:     end if
0:   end if
0: end for
0: endproc =0
```

Algorithm 1 shows the approach example [5] . At first, initially this algorithm starts with the sorted node list and the variables n, *depth* and m. Besides, the nodes with no edge will not recognize. Variable n contains the number of nodes in the list. *Depth* is set to 1 by *max_clique* . Furthermore, *depth* is the recursion counter of the algorithm. Variable m is set initially to 0. Whenever a clique is found m is increased by 1. Let $v_1 v_n$ the sorted list of nodes without the nodes with no edges. Algorithm 1 pass the list from left to right and tries to find the maximum clique C_1 containing node v_1. Following the maximum clique C_2 will be found on the graph *G*- v_1 that contains v_2. In the next step the maximum clique C_2 containing v_3 will be searched on the remaining graph *G*-$v1$, $v2$,etc. . The algorithm operates on adjacent matrices.

All nodes adjacent to v_1 are stored in list L_{v_1} . The variable depth is set to 2 by the call of *max_clique*. The node list L_{v_1} are passed beginning with the first node v_1. Then, a list L'_{v_1} will all adjacent node to v_1 is generated. Hence, these nodes are also adjacent to v_1.

Besides that, as example which depicts in "Fig. 9" , by applying this algorithm , 2 clique can be found as shown in "Fig. 9" since there is no adjacent nodes to $P_{src,des,5}$. So, based on the algorithm, maximum clique is now 2 which are $P_{src,des,3}$, $P_{src,des,5}$ Therefore, this clique can be removed by doing a partition.

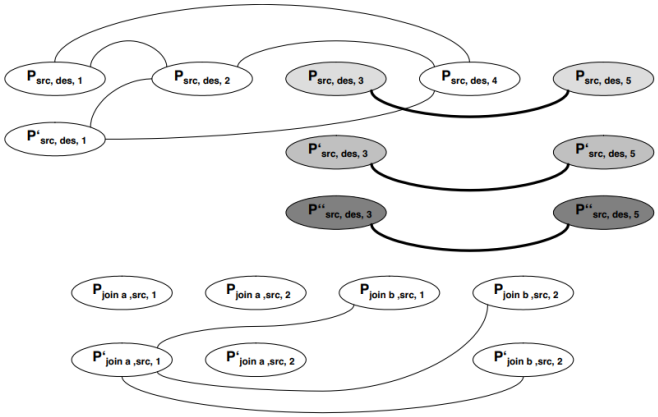

Fig. 9. 2 cliques are found based on algorithm 1 on example of compatibility graph [1].

Later, clique algorithm will be applied at compatibility graph. Example of compatibility graph will be illustrated in "Fig. 9". In the example, the compatibility graph consists of 17 nodes resp. paths and 11 edges. As discussed earlier, the main target is to find clique on the graph.This reduces the power consumption by combining the nodes that are in a clique into one control partition control. In the next section, example model and experimental result will be discussed more details how the power consumption can be minimized and optimized in a scheduling process.

E. Example Model Experimental Result

In this topic, the design example of scheduling under low power constraint will be discussed more details.

Based on dataflow graph as depicted in "Fig. 10" ,power consumption is identified for every node. This follows :

Definition : (limited scheduling respectively to power consumption) Find a schedule that fulfill the following formula:

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

Whereas m is indicates number of real resources of a certain type in a schedule s, pi is power consumption of a real resource and duration Di of node I within a schedule$s$. Besides,

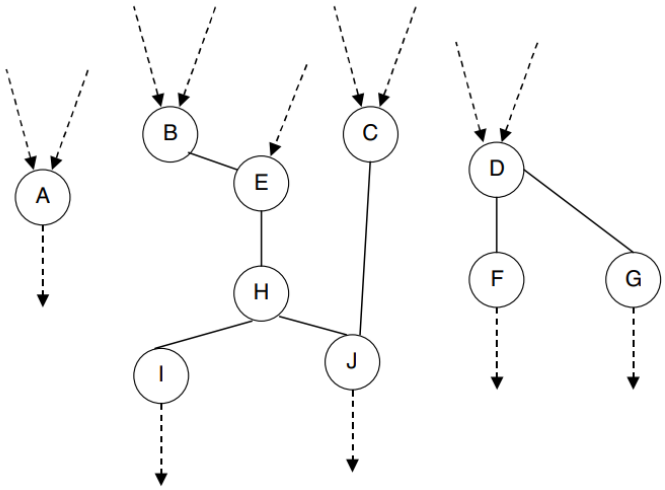

Fig. 10. Example of Dataflow Graph [1]

g_s is the additional cost for switch on/off mechanism for a schedule s. In the "Fig. 10", 10 operators are included with 3 different partitions. Every partition can be either switched on or off. The first partition contain only operation A while the second partition have 5 operations which are B, C, E, H, I and J. The last partition have 3 operations which are D, F and G.

Based on the divided partitions, it is clearly shown it is just a waste of power if C operation is running for the entire running time. However, to get the idea behind this, mobility calculation is necessary to describe the execution timing interval of a node. This can be done by a computation of a As Soon As Possible (ASAP) and As Late As Possible (ALAP) scheduling. For ASAP, the execution time for the operations will be the earliest time while for ALAP, the running time is opposite to ASAP which is the latest point of the time. Therefore, mobility can be computed as below :

$$Mobil_v = alap_v - asap_v + 1 \tag{3}$$

As shown in "Fig. 11", example of data flow graph by using ASAP scheduling algorithm is described. Besides that, "Fig. 12" depicted how ALAP algorithm works in data flow graph. Both of the figures ASAP and ALAP, we can figure the time interval required for the operations to be executed for the data flow graph as depicted in "Fig. 13".

Hence, the mobility for each node by using equation "mobility" is as below :

- MobilA = 4
- MobilB= MobilE= MobilH= MobilI= MobilJ = 1
- MobilC = MobilD = MobilF = MobilG = 3

To achieve low power consumptions in scheduling, calculation of mobility is vital, so that operations that have mobility greater than 1, could be executed in different time frames.

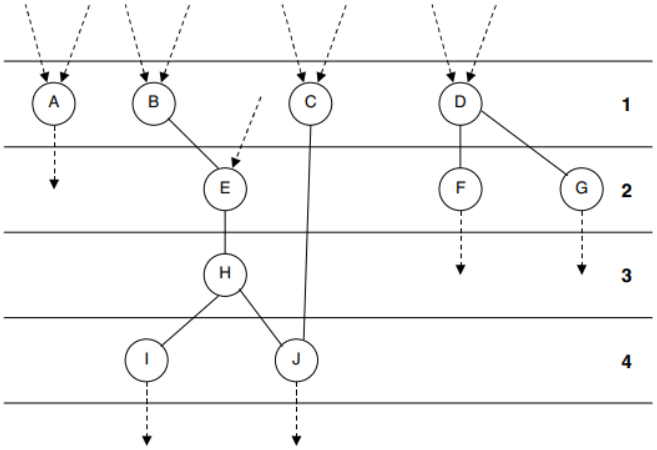

Fig. 11. Example of Dataflow Graph with ASAP schedule [1]

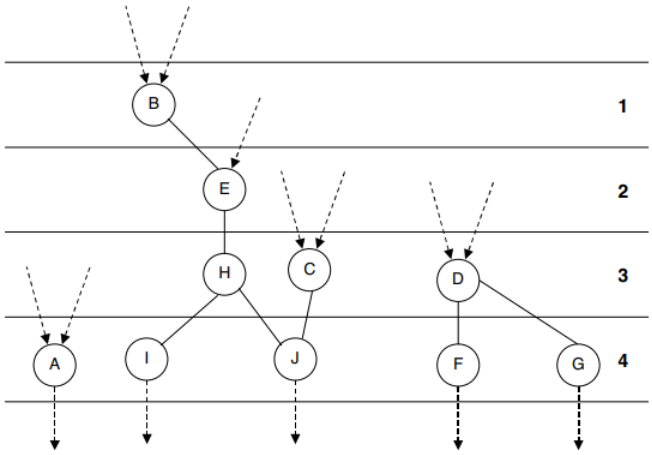

Fig. 12. Example of Dataflow Graph with ALAP schedule [1]

Therefore, the operations can be switched on only when the operation is required to execute and the operations can be switched off if it is not required [1].

Besides that, as discussed earlier, by doing partitions, power consumption can be minimized. However, it is important to take into account that the delay, the area for each resource, the power consumption and the data dependencies. For sure, power can be lower if reducing the real resources, however it will causes lots of challenges in mapping the operations to the same real resources. To solve this issue, by including registers and multiplexers in design process is the good approach but it will consume more power. Therefore, activation and deactivation mechanism like gated clock and guarded evaluation is better than saving real resources that can cause more complicated works [1].

Based on information in "Fig. 13", operation A is an independent operation and can be executed at any of the 4 cycles since operation A is the only operation in its partition and has mobility= 4. That means, it can be activated or

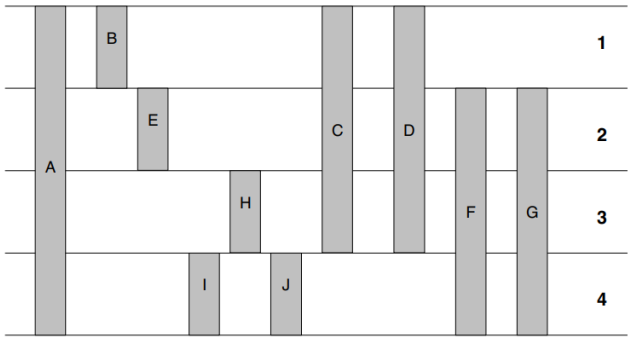

Fig. 13. Execution time intervals of operators from Dataflow Graph [1]

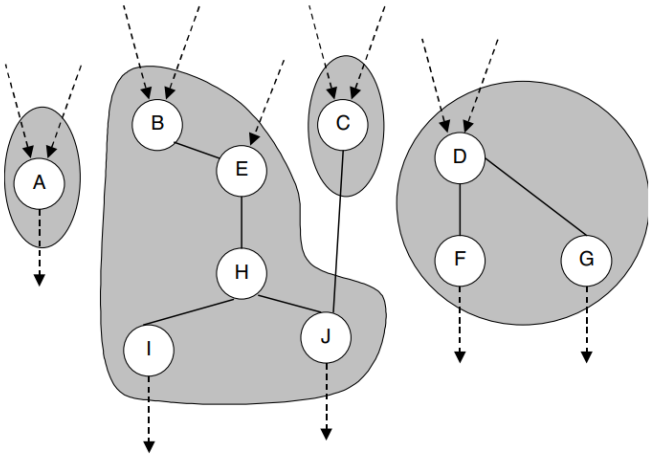

Fig. 14. Partition of operators as example of methods to gain low power [1].

deactivated without any consequences to other operations. Since operations B, E, H, I and J have only 1 mobility, these operations are excluded from this analysis but there are group in the same partition so that, it can be executed at the assigned time frame. For operation C, it has mobility= 3, so the same case as operation A, it can have its own partition, so that it can be executed at any of its 3 cycle of the execution time but it must be before execution time of operation J. By doing this, operation C can be deactivated for 2 cycles. As a result, lower power is consumed since operation A and C that can be switched off at unused time frames. The explained partition is depicted in "Fig. 14". Gated clocks or guarded evaluation can activate or deactivate every partition that has created.

This strategy is based on the data dependencies of the operations being recognized from the dataflow graph. The execution intervals and mobility are then calculated using the ASAP and ALAP schedules. The activation and deactivation partitions are computed using the mobility and data dependencies as inputs.As depicted in "Fig. 15", First results of different small filter algorithms for compression shows, that it is possible to achieve up to 20% minimization of the power consumption. For the standard high-level synthesis benchmark "differential equalizer" it is possible to save up to 10% of

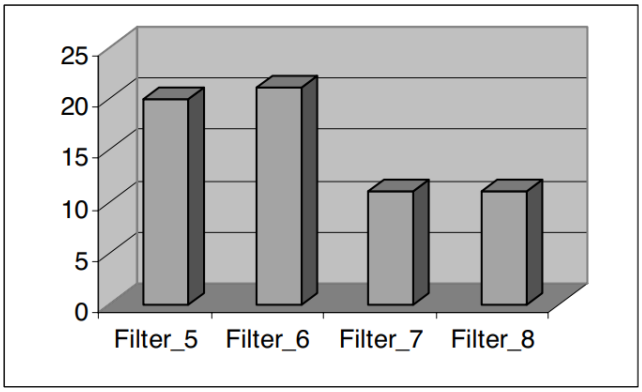

Fig. 15. Results of pwer saving for difference filter algoritms [1]

the power consumption by using the low power high-level synthesis for a bit-serial design compared to regular bit-serial implementation.

VI. Conclusion

In this paper, lots of methods and strategies are presented to optimise hence, reduce the lower consumption in scheduling process. By switching activity, finding cliques, doing partition and many other strategies can be implemented in scheduling process to optimise the power efficiently. Furthermore, scheduling process is one of important levels in High Level synthesis. Since there is big amount of power usage in this technology industry, low power scheduling plays a big role to help in this matter.Therefore, power and energy can be saved for the future.

VII. Acknowledgement

I would love to express my special thanks to Prof. Dr. Achim Rettberg for the guidance and support that has given in making this paper.

References

[1] [1]A. Rettberg, B. Kleinjohann and F. Rammig, "Integration of Low Power Analysis into High-Level Synthesis", Design and Analysis of Distributed Embedded Systems, 2002. Available: 10.1007/978-0-387-35599-3-20 [Accessed 18 June 2022].

[2] G. De Micheli, Synthesis and optimization of digital circuits. New York, N.Y.: McGraw-Hill, 1994.

[3] J. Kun-Lin Tsai, Ju-Yueh Lee, Shanq-Jang Ruan and Feipei Lai, "Low power scheduling method using multiple supply voltages," 2006 IEEE International Symposium on Circuits and Systems (ISCAS), 2006, pp. 4 pp.-, doi: 10.1109/ISCAS.2006.1693828.

[4] S. Mohanty, Low-power high-level synthesis for nanoscale CMOS circuits. New York: Springer, 2008.

[5] Randy Carraghan and Panos M. Pardalos. An exact algorithm for the maximum clique problem. In In Operation Research Letters 9, pages 375–382, 1990