

Low Power Driven High-Level Synthesis for Dedicated Architectures

Achim Rettberg, Bernd Kleinjohann, Franz Rammig
University Paderborn / C-LAB
Fuerstenallee 11, 33102 Paderborn, Germany
Tel: +49 5251 606110, Fax: +49 5251 606065
Email: {achim, bernd, franz}@c-lab.de

Abstract

This paper describes a method to integrate low power scheduling and partitioning into high-level synthesis. The high-level synthesis approach is especially developed for a bit-serial architecture. We addressed a specific analysis technique within the scheduling task of high-level synthesis. The analysis technique allows the determination of dedicated turn-on and turn-off mechanism by design partitioning. Therefore, the optimisation of power consumption is simultaneously improved with the design delay for the target architecture.

1 Introduction

Today highly integrated circuits and components entered all design areas. Especially consumer electronic devices, such as mobile-phones and PDA's belong to those integrated circuits. This battery driven devices don't have the ability to recharge the batteries at any time. For this reason, the usage of low power components plays a major role to receive longer operation time of these devices. The primary objective of such developments is the minimization of power consumption. To receive this objective, some methods are used to optimise the devices operation by adjusted cycle frequencies or deactivation of external devices and displays. Other methods optimise the power consumption of devices without modifying the functionality. Usually in this area it is expected to save power consumption. For example, at a laptop 52 % power is needed for the motherboard. The display consumes only 8 % [1]. Methodological investigation shows that there exists in particular asynchronous architectures methods to estimate power consumption [2], [3], [4], [5]. The power consumption is often lower by two sequenced signals with the same value in

opposite to signal switch with different values. Sequenced signals with the same value occurred often, because in synchronous designs each rising and falling edge generates an output value. Different estimation methods based on these techniques are developed [6], [7]. If we embed this methods in the methodological design of digital systems, a bottom-up approach will be recognizable. Optimisations referred to elementary cells of the final implementation and the data encoding are presented in [8], [9] and [10]. In generally power can be saved on register-transfer level by consideration of different architecture variants [7]. Furthermore it is possible to introduce parallelism and pipelining on architectural level to decrease power consumption. Another possibility offers the usage of guarded evaluation. The integration of registers on the primary inputs of a logical block avoid the switching of it. These registers store the values if the logical block is not used. The insertion of so called gated clocks allows the turning off of non-active design parts [6], [11]. Especially techniques like operand isolation and pre-computation decreases the power consumption enormously. "Operand isolation" means that the operands are computed only once. The splitting of a calculation in pre- and main calculation is called pre-computation. But the integration of those techniques into the synthesis process is very complex and requires a substantial effort. The selection of the design paradigm (synchronous, asynchronous) influences on one side the power consumption and on the other side the implementation. Furthermore, the methods used for the optimisation are described in [12], [13]. Asynchronous architectures are good for low power designs, because hereby exists no clock-signal within the design. The asynchronous components are

self-synchronizing by a handshake-mechanism [14]. Thus only active parts of an asynchronous design consumes power. The introduction of parallelism into the asynchronous architecture leads not to a increasing power consumption. The realization of the handshake-mechanism is a design effort. That mean, we need more wires for the implementation. This has to be considered with respect to the asynchronous architecture in opposite to a synchronous implementation.

In this paper we present a method, that discussed the optimisation of the power consumption on the architecture level for dedicated bit-serial architectures. The method starts from a dataflow graph. We take into account the analysis method for asynchronous bit-serial architecture that is presented in [15]. In the described method, that is integrated in the high-level synthesis, we develop an analysis method of architectures based on a dataflow level for bit-serial and bit-parallel architectures. The determined algorithms are filter-algorithms, which are used for signal pre-processing.

2 Related Work

In the past several algorithms for high-level synthesis are developed. The major objective for all these algorithms was the minimization of the used resources to save chip area and the optimisation of the systems delay time. An interesting approach for the integration of low power techniques into the high-level synthesis is presented in [8]. The approach focuses on the minimization of resources per cycle whereby power is saved. This could be achieved by mapping the same operation types to a real resource.

Another interesting approach, that performs a power estimation for behavioural level is presented in [16]. The behavioural models are implemented in VHDL. Furthermore, parallelism and pipelining can be used on architectural level to decrease the power consumption.

The problem of minimizing power dissipation during various stages of behavioural synthesis is addressed by some researchers. In [17] they uses pipelining and parallelization as behavioural transformations that compensate the delay increase due to the use of reduced supply voltage levels.

3 Methodology and analysis

Generally, the design of a digital system bases on a high-level specification. The specification is transformed into an algorithmic description, like C or behavioural VHDL source code. During the high-level synthesis the behavioural description of the algorithm is compiled into a structural description. Within the high-level synthesis several method are performed to realize the synthesis process. The methods are namely scheduling, allocation and binding. The scheduling process organizes the operation according the timing information. The mapping of the temporally organized operations and memory elements to real resources is done in the allocation and binding processes. In past approaches, these methods optimises only the delay time and the used area of the digital system. In this context the minimization of the power consumption isn't considered. In our approach, the minimization of the power consumption is the main objective.

Therefore, results of the activation interval analysis [14] based on a special asynchronous architecture [12], [13] should be integrated into the high-level synthesis. The advantages of bit-serial architectures are the elimination of size overhead on logic level for each operation and that parallelism kept upright by using pipelining design style. During the high-level synthesis the algorithmic description is transformed into an internal format, called dataflow graph. The nodes in the dataflow graph correspond to the operations of the algorithmic description. The edges indicate the data dependencies between the operations. The following definition gives a formal description of the dataflow graph.

Definition 1: Given a directed dataflow graph $G = (V, E)$ with the set $V = \{v_1, \dots, v_n\}$ of operations within the graph and the set $E = \{e_1, \dots, e_m\}$ for the data dependencies.

Complex operations within the dataflow graph, such as trigonometric functions are replaced by the corresponding algorithm. Each node in the graph is annotated with some characteristics, as power consumption, number of control signal and timing information. The timing information consisted of the delay and throughput. Along the edges we can observe

the activation of the specific operations [14]. In our bit-serial asynchronous architecture, during the analysis phase, each operation-node corresponds to a real resource. Therefore, for the asynchronous architecture it is not necessary to perform optimisation on the dataflow graph. But, if we want to analyse bit-parallel architectures an optimisation of the operations within the dataflow graph is important and necessary. This means, we map the dataflow graph to a set of real resources $M = \{m_1, \dots, m_k\}$. This will be carried out by the already mentioned high-level synthesis. A formal definition of a valid implementation after the synthesis process is given in definition 2.

Definition 2: Given a graph according to definition 1 and a set $M = \{m_1, \dots, m_k\}$ of real resources. For each resource m_i the timing t_i is known. Furthermore a latency-bound L is given. A valid implementation that consists of scheduling, allocation and binding has to fulfil the following terms:

- The latency-bound must be strictly adhered, that mean the delay time of the system is below L .
- The number of nodes running simultaneously on instances of a resource type is lower than the number of allocated instances of the type.

This definition shows the main tasks of the high-level synthesis. Obviously, an analysis of the power consumption could be integrated during the scheduling process. As mentioned before, the main objectives of the high-level synthesis are the minimization of run-time and chip area of the system. An integration of the power consumption optimisation process based on execution interval analysis is described in the next chapter.

4 Scheduling under low power constraints

Without loss of generality we assume that the power consumption p_i is known for each node v_i within the dataflow graph. From this follows:

Definition 3: (limited scheduling respectively to power consumption) Find a schedule that fulfil the following formula:

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

Whereas m_{is} is a number of real resources of a certain type in a schedule s , power consumption p_i of a real resource and duration D_i of node i within a schedule s . Furthermore, g_s represents the additional cost for switch-on/off mechanism for a schedule s .

Therefore, the value α defines a bound for the scheduling algorithm. The activation interval analysis described in [14] calculated three different partitions for the dataflow graph depicted in Figure 1, which contains 10 operators. Each partition can explicitly controlled, that mean it can switched on and off. The first partition contain only the operation A. The second one consists of B, C, E, H and J. The last and third partition consists of D, F and G.

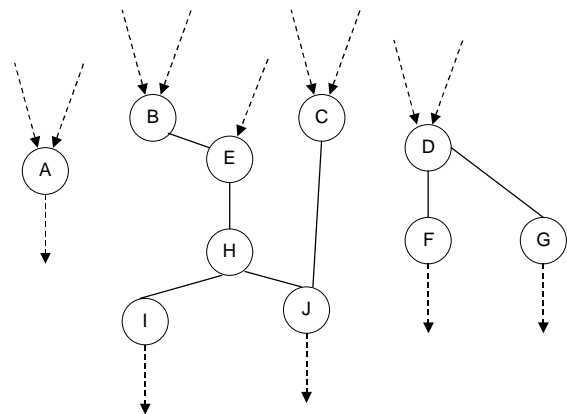


Figure 1. Example of a dataflow graph

Regarding node C in the second partition of the example dataflow graph, it is obvious that it is not necessary to activate this node for the entire running time. The calculation of the so called mobility, which describes the execution timing interval of a node, can be done by a computation of a ASAP¹ and ALAP² scheduling. These scheduling methods arrange the nodes of the dataflow graph according to the earliest respectively to the latest point of

¹ ASAP = as soon as possible

² ALAP = as late as possible

time. The mobility can be computed by the execution timing interval $[asap_v, alap_v]$ for each node v of the dataflow graph. The formal definition is:

$$Mobil_v = alap_v - asap_v + 1$$

It is possible to compute both schedules in linear time. The pseudo-code of the scheduling algorithm is depicted in Figure 2.

```

Public static DesSpace optimate() {
    . declaration & initialisation
    .
    // E_a is actual node in graph, for loop
    // prevDS allways contains last found
    // solution
    // starts with prevDS is ASAP
    while (E_a.next != null) { E_a = E_a.next;
        Copy old graph for backtrack

        find first occurrence of E_a in ASAP-
        schedule
        determine number of Entities with same
        Type like E_a as #E_a

        find SCD_Drn as next time in schedule,
        with less Entities of same Type
        as E_a than #E_a
        (findNextDrain)

        push E_a to SCD_Drn and correct Graph
        with new values of E_a,
        following the dependencies on
        E_a and followers
        (drift)

        create an schedule newDs and estimate
        the new power consumption
        if newDS better than prevDS {
            keep new schedule in prevDS
            restart loop with first Entity in Graph
        } else {
            dismiss newDS
            reset graph to previous values
        }
    }
    One last loop for improvement of the found
    solution
    return actDS;
}
    
```

Figure 2. Scheduling algorithm

At the beginning the ASAP schedule is calculated. Furthermore, the function “findNextDrain” calculates the ALAP schedule, followed by the calculation of the mobility of operations (see function “drift”).

The next step is the estimation of the power dissipation by usage of a backtracking.

The ASAP schedule is depicted in Figure 3 for the example dataflow graph. Figure 4 shows the ALAP schedule of the example.

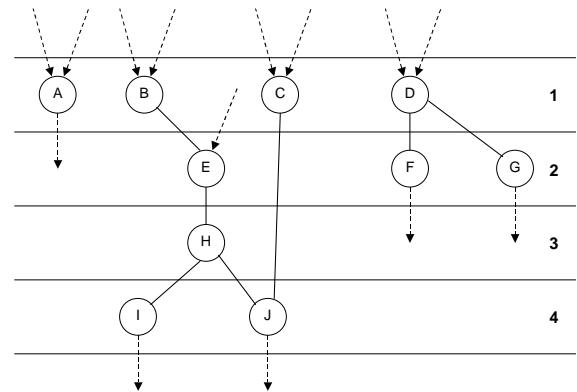


Figure 3. Dataflow graph according to the ASAP schedule

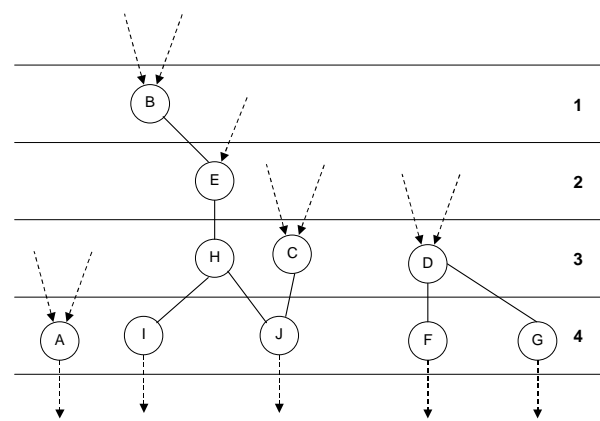


Figure 4. Dataflow graph with ALAP schedule

combined with others to maintain a guarded A graphical representation of the execution timing intervals for the full design is depicted in Figure 5. Therefore, the mobility for each node is:

- $Mobil_A = 4$
- $Mobil_B = Mobil_E = Mobil_H = Mobil_I = Mobil_J = 1$
- $Mobil_C = Mobil_D = Mobil_F = Mobil_G = 3$

The operations with mobility equal 1 are fixed for exactly one time frame in which they are executed. Operations with a mobility grater than 1 could be executed in different time frames. Especially, those operations are

relevant for the minimization of the power consumption. That means, they can explicitly accessed by a switching on and off mechanism. Therefore, they contribute to a low power design. Besides this, those operations can be partition, but it is necessary to take into consideration the delay, power consumption, area for each resource and the data dependencies. This is reasonable for bit-serial architectures with a high data bit-width, because real resources could be saved. Power are saved by the minimization of real resources, but the mapping of operations to the same real resource increases the communication effort. For this reason, it is necessary to include registers and multiplexers in your design, but such components consumes additional power. In the following, we assume not to save real resource, but to integrate activation and deactivation mechanism like gated clocks and guarded evaluation [11].

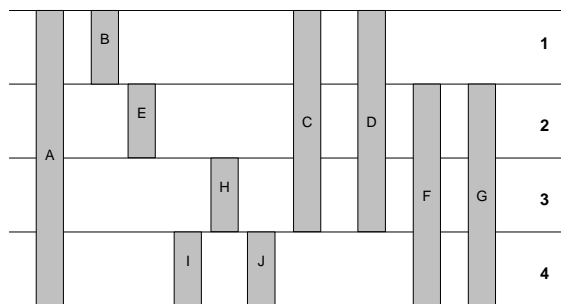


Figure 5. Execution intervals of the operators from the dataflow graph

If we summarized the information about the data dependencies and the mobility it is observable that operation A is completely independent of all others. Moreover operation A can be executed at any of the 4 cycles. Additionally, these operation could form an own partition, which can be explicitly activated or deactivated, but the aim is to combine operations that are active/passive in the same time frame to a guarded partition. In accordance with the previous discussion, operations D, F and G can be executed within 3 cycles. As mentioned before, operations B, E, H, I and J have the mobility 1 and will not be considered for the analysis, but those operations build another partition. At this point we received with this method the same results as from the activation interval analysis [14]. Obviously operation C isn't assigned to a

partition. The mobility of operation C is 3. For this reason, it is necessary to execute C before the computation of operation J is started. This has to be done within the first 3 cycles. Therefore, operation C can be deactivated for 2 cycles. Basically, this contributes to the minimization of power consumption which could not be identified with an activation interval analysis. One possible partition of the complete dataflow graph is depicted in Figure 6. Each partition can be activated or deactivated with gated clocks or guarded evaluation. If gated clock are used the additional costs based on four AND gates (one for each partition). As mentioned before the target architecture for the high-level synthesis is a bit-serial design. Therefore, the size of a real resource is in most cases smaller than multiplexer and register that are used by mapping of operations to a real resource.

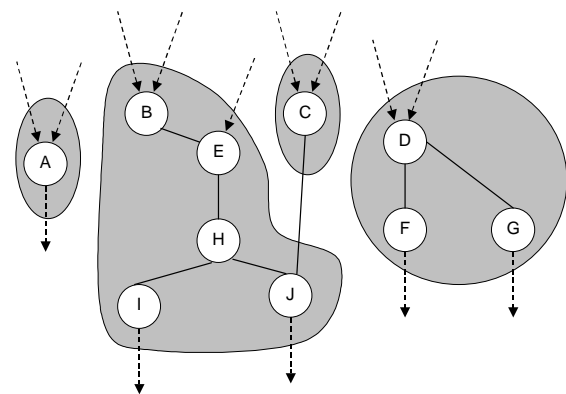


Figure 6. Partitioned dataflow graph

This method depends on the recognition of the data dependencies of the operations from the dataflow graph. Afterwards the ASAP and ALAP schedules are computed to calculate the execution intervals and the mobility. In combination with the mobility and the data dependencies the activation and deactivation partition are calculated. First results of different small filter algorithms for compression shows, that it is possible to received up to 20 % minimization of the power consumption, see Figure 7. For the standard high-level synthesis benchmark “differential equalizer” it is possible to save up to 10 % of the power consumption by using the low power high-level synthesis for a bit-serial design compared to regular bit-serial implementation.

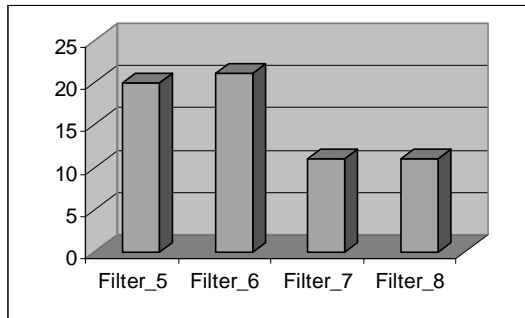


Figure 7. Power savings in percentage for different filter algorithms

5 Conclusion and Future Work

This paper describes an analysis method based on a dataflow graph that allows the integration of control mechanism for low power designs into the high-level synthesis. Especially, the scheduling which is part of the high-level synthesis is used to activate only active operations.

Furthermore, it is planned to include the allocation and binding processes in the low power analysis. In addition to this, the implementation of a comprehensive design space exploration toolkit focuses on the aspects of power consumption so that timing behaviour is planned.

References

- [1] M. J. Irvine, "Low Power Tutorial", in Proceedings ASIC/SOC, Washington D.C., 1999
- [2] M. Laurent, M. Briet, "Low Power Design Flow and Libraries", in NATO ASI Series "Low Power Design in Deep Submicron Electronics", Kluwer Academic Publishers, 1997
- [3] P. Kudva, V. Akella, "A technique for estimating power in self-timed asynchronous circuits", In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems November, 1994
- [4] J. Tierno, R. Manohar, M. Martin, "Energy and entropy measures for low power design". In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, März, 1996
- [5] P.A. Beerel, C.-T. Hsieh, S. Wadekar, "Estimation of energy consumption in speed-independent control circuits". In International Symposium on Low Power Design, 1995
- [6] S. Devadas, S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits". In Proc. of the 32nd Design Automation Conference, San Francisco, CA, Juni, 1995
- [7] C. Piguet, "Circuit and Logic Level Design", in NATO ASI Series "Low Power Design in Deep Submicron Electronics", Kluwer Academic Publishers, 1997
- [8] J. Monteiro, S. Devadas, B. Li, "A methodology for efficient estimation of switching activity in sequential circuits". In Proc. of the 31st Design Automation Conference, San Diego, CA, 1994
- [9] M. Stan, W. P. Burleson, "Bus-Invert Coding for Low Power I/O", IEEE Transactions on VLSI Systems, 1995
- [10] Koegst, M.; Franke, G.; Rülke, St.; Feske, K., "A Strategy for Low Power FSM-Design by Reducing Switching Activity", PATMOS '97, Sept. 8-10, 1997, Louvain-la-Neuve, Belgium
- [11] L. Benini, G. De Micheli, "Transformation and Synthesis of FSMs for low-power gated-clock implementation". IEEE Transactions on Computer-Aided Design, 1996
- [12] W. Hardt, B. Kleinjohann, "Flysig: Towards High Performance Special Purpose Architectures by Joining Paradigms", in Proceedings of 7th NASA Symposium on VLSI Design, Albuquerque, New Mexico, Oct. 1998
- [13] W. Hardt, B. Kleinjohann, A. Rettberg, L. Kleinjohann, "A New Configurable and Scalable Architecture for Rapid Prototyping of Asynchronous Designs for Signal Processing", in Proceedings of the 12th Annual IEEE International ASIC/SOC Conference, Washington, DC, USA, September 1999
- [14] P.B. Endecott, "SCALP: A Superscalar Asynchronous Low-Power Processor", Dissertation, University of Manchester, 1995
- [15] 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
- [16] Kruse, L.; Schmidt, E.; Jochens, G.; Stammermann, A.; Nebel, W. "Lower Bounds on the Power Consumption in Scheduled Data Flow Graphs with Resource constraints", In Proc. of the Design and Test Conference in Europe (DATE 2000), Paris, France, 2000
- [17] A. Chandrakasan, M. Potkonjak, J. Rabaey, R.W. Brodersen, "HYPER-LP: A System for Power Minimization Using Architectural Transformations", in Proceedings of IEEE International Conference on Computer-Aided Design, 1992