Power Constrained High-Level Synthesis of Battery Powered Digital Systems

S.F. Nielsen and J. Madsen
Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads, Bldg. 322
DK-2800 Kgs. Lyngby, Denmark
{sfn,jan}@imm.dtu.dk

Abstract

We present a high-level synthesis algorithm solving the combined scheduling, allocation and binding problem minimizing area under both latency and maximum power per clock-cycle constraints. Our approach eliminates the large power spikes, resulting in an increased battery lifetime, a property of outmost importance for battery powered embedded systems. An interesting side effect of our approach is that it often obtains a smooth power profile, which effectively contributes to the increased battery lifetime. Our approach extends the partial-clique partitioning algorithm of [3] by introducing power awareness through a heuristic algorithm which bounds the design space to those of power feasible schedules. We have applied our algorithm on a set of dataflow graphs and investigated the impact on circuit area when applying different power constraints.

1. Introduction

Our target applications are low-cost portable embedded systems. Today, consumers demand portable applications so tiny that they go virtually undetected when not in use. This puts several severe constraints on the system eg. there is little space for batteries. At the same time consumers require the latest features and top quality of the product, this means additional digital signal processing, as well as a general increase in computation load. An interesting aspect of this application area is the low-cost issue which puts focus on reducing the overall system cost, eg. a requirement to select a low-priced (low-quality) battery over a high-priced (high-quality) battery.

Now, the amount of total energy/charge available from a battery, and thus its life-time, depends strongly on the current profile of the application [1, 2]. Here there are two contributing factors: (1) If the peak-current exceeds a maximum-threshold the life-time starts dropping dramatically. (2) A large current variation also leads to reduction in battery life-time. These factors are more dominant on batteries of low quality, where up to a 20-30 percent extension of life-time has been reported when designing for battery powered systems [1]. In figure 1 (top) is shown an undesirable schedule which violates the power constraint as well as having large variations. Below is shown a more desirable schedule which fulfills our power constraint and has less variations. Here we will primarily focus on the first factor, leading to a maximum power usage per clock-cycle constraint, which has shown to be the dominating factor [1].

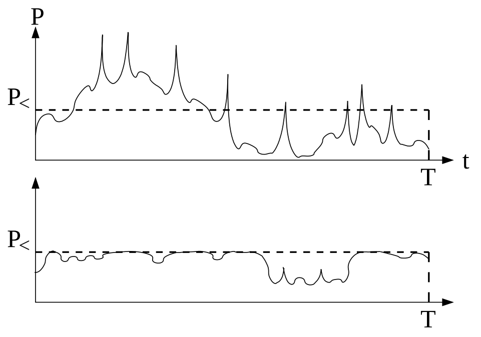


Figure 1. Undesired power schedule (top). Desired power schedule (bottom).

Naturally, power consumption is of greatest concern here but for this low-cost application area reducing the used silicon area is still a very important issue. Our goal is therefore to reduce the area of the digital circuit given an application which has to complete within a hard time constraint and a maximum power usage per clock-cycle constraint.

So far the main efforts in low-power synthesis can be divided into two groups: *(a) Low level allocation and assignment* [4, 5, 6, 7]: Here the goal is to combine functional units and operations in such a way the the internal switching activity of the functional units is minimized. *(b) Task level scheduling* [1, 2, 8]: Here the goal is to schedule tasks in such an order the peak system power is minimized. The majority of these algorithms are either based on meta-heuristic algorithms, or two-step algorithms [1, 2] where in step one a traditional time constrained schedule is constructed and in step two the schedule is made "power-aware".

In this paper we position our selves in between and present a heuristic synthesis algorithm which solves: (i) Scheduling, (ii) Allocation and (iii) Assignment simultaneously under both a time and power constraint. These 3 tasks are traditionally solved separately which is suboptimal as these typically interfer with each-other. This enables us to expand the exploration of the design space to include different types of functional units eg. the speed and energy usage of an operator can be traded versus the area of the operator.

The outline of the paper is as follows: first a heuristic power-scheduling algorithm is presented in 2 and then in section 3 the central power and time synthesis algorithm is presented. In section 4 we present the benchmark tests we have performed and in section 5 we elaborate on our results. Finally, the conclusion is presented in section 6.

2. Power Heuristic Scheduling

In traditional time constrained synthesis the two heuristic low complexity algorithms, **asap** and **alap**, are used to bound the solution space. In figure 2 is shown an example CDFG, which we will use throughout this paper, and its corresponding **asap** schedule, where we have assumed all operations without loss of generality are executed in one cycle.

In the following we will present a heuristic algorithm, **pasap**, which given a power constraint generates a schedule. This algorithm will play the same role as **asap** and have been used in our main algorithm to heuristically bound the minimum time separation between two operators, ensuring all CDFG precedence constraints are satisfied together with the power constraint. The main idea behind this algorithm is to "stretch" the **asap** schedule to fit the power constraint, ie. to schedule the operators as fast as possible, but only if there is power available meaning some operators will be delayed additional cycles.

pasap ($P_<$):

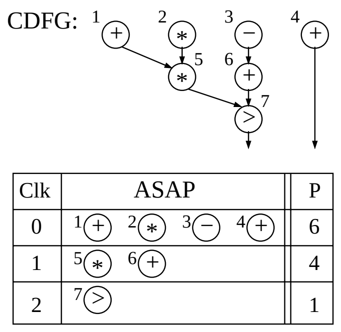
Initialize: Schedule source start-time to zero and initialize the execution offset o_i (cycles) to zero for all operators.

step 1: Pick an unscheduled operator v_i

step 2: If v_i has unscheduled predecessors, goto 4.

FU	σ	delay(clk)	Area	Power
ADD	$\{+\}$	1	1	1
ALU	$\{+,-,>\}$	1	1.5	1
Mult	$\{*\}$	1	4	3

Table 1. Simple example FU library.



Clk	ASAP	P
0	$^1(+)$ $^2(*)$ $^3(-)$ $^4(+)$	6
1	$^5(*)$ $^6(+)$	4
2	$^7(>)$	1

Figure 2. Example CDFG and its asap schedule

step 3: If there is power available in the execution time interval $[(t_i+o_i)..(t_i+o_i+d_i)]$, where d_i is the execution delay of v_i and $t_i = \max\{t_j + d_j\}\ \forall v_j \to v_i$, is the earliest start time, otherwise increase o_i by one.

step 4: If unscheduled operators, goto step 1.

For construction of our **pasap** schedule we will use the simplistic functional unit (FU) library shown in table 1. In figure 3 is shown the **pasap** schedule for our example CDFG, here we have set a power limit of $P_< = 3$, which we will keep for this example. The algorithm starts in cycle one and tries to fill it up with operations: we start by scheduling v_1, which prevents us from scheduling v_2 as this would violate the power constraint. But we can continue to schedule v_3 and v_4. In the next cycle we have v_2 ready, which is the only one for which there is power available and the algorithm continues. The total **pasap** schedule takes 5 cycles to complete as opposed to only 3 cycles of the **asap** schedule. The same algorithm can run backwards which we will denote **palap**.

Obviously there are many ways of selecting which operators to "pack" into cycles and it is a hard problem to find the optimal combination ie. the one results in the schedule using the least amount of time, here we have simply chosen the order of which they appear in the CDFG. In this way **pasap** cannot be compared to **asap**.

Clk	PASAP	P	PALAP	P
0	$^{1}\oplus$ $^{3}\ominus$ $^{4}\oplus$	3	$^{2}\circledast$	3
1	$^{2}\circledast$	3	$^{3}\ominus$ $^{1}\oplus$	2
2	$^{5}\circledast$	3	$^{5}\circledast$	3
3	$^{6}\oplus$	1	$^{6}\oplus$	1
4	$^{7}\textcircled{>}$	1	$^{4}\oplus$ $^{7}\textcircled{>}$	2

Figure 3. The pasap and palap schedules of our example CDFG, both with P$_<$=3.

3. Power and Time Constrained Synthesis

In figure 4 we have re-shown our example CDFG as well as a non-power constrained schedule with a time constraint of T=5 cycles. Here the partial clique partitioning algorithm in [3] is capable of constructing a schedule and an FU allocation using only one ALU and one Mult (the minimal FU-allocation to execute this CDFG no-matter how much time we have available) using a total area cost of 5.5 units. Now alongside the schedule is shown the power consumption in the respective cycles. Here we note two things: (i) This schedule violates the power constraint of $P_< = 3$ and furthermore (ii) it is very spiky (cycles 1 and 3). For a power constrained schedule we wish to stay under our constraint and "smoothen-out" the schedule.

As mentioned, our power constrained synthesis algorithm builds upon this algorithm and as in [3] we will construct the time-extended compatibility graph, $V1$: Each vertex A_{ijk} represents a possible scheduling, allocation and assignment of operation i on FU type j starting in cycle k. Each edge $< A_{ijk}, A_{rjt} >$ represents the simultaneously scheduling, allocation and assignment of operator i and r on the same FU instance of type j at times k and t, respectively. We have extended the formulation of a valid $V1$ graph to include power constraints. Thus our allowed vertices (A_{ijk}) are:

i: All operators in the CDFG.

j: The set of FUs where operator i can be executed.

k: The time interval given by $\{t_{\mathbf{pasap}}, t_{\mathbf{palap}}\}$, when operator i is executed on FU j and all other operators are scheduled using delay information from the fastest FU type and power information from the most power hungry FU type.

And the allowed edges,$< A_{ijk}, A_{rjt} >$, are those where there is a dependency in the CDFG, $v_i \rightarrow v_r$, and the execution time of the two operators does not overlap when scheduled on FU_j, as well as it is possible to find a valid pasap schedule with v_i and v_r scheduled on FU_j at times k and t respectively.

A subgraph of $V1$ which is completely connected by compatibility edges in $V1$ (clique) can be mapped to one FU instance. Then the solution to the synthesis problem with the minimum area and using least interconnect is the problem of finding the **Partial minimal cost clique partitioning of $V1$ which does not violate the power constraint**, where partial refers to a cover containing one-and-only-one vertex for each operator i.

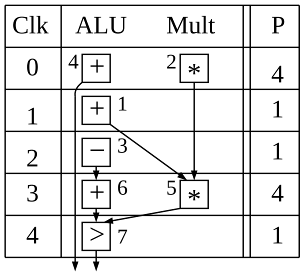


Figure 4. CDFG and a non-power schedule with T=5, using only one ALU and one Mult with a total area of 5.5.

As in [3] we will heuristically solve the clique partitioning problem, through a greedy approach ie. evaluate the $V1$ graph and pick a "best" decision which is then scheduled, allocated and assigned and then repeat the process until no operators are left. To this end we will construct the Mixed-vertex Compatibility Graph ($MCG = (V1, V2, E)$): The $V1$ graph, extended with super-vertices $S_{j,n} \in V2$. The super-vertexes $S_{j,n}$ contain the scheduled, allocated and assigned operators on FU of type j instance n. Initially $|V2| = 0$.

In principle, our algorithm starts with a power and time valid region then aggressively reduces area ensuring the scheduling region stays valid. Our algorithm is as follows:

Preparatory Step: *Build the MCG*. Here **pasap** and **palap** are used to build the set of allowed vertices and allowed edges, under the power and time constraint.

Step 1: *Pick the best decision*. We select according to maximum clique: Find the largest clique A_{ijk} is contained in (a double search of the entire graph). $cost_{A_{i,j,k}} =$ sum of FU area for maximum clique($A_{i,j,k}$). The selected vertex is merged into an existing super-vertex if it is connected to a super-vertex, otherwise it is made into a new super-vertex.

Step 2: *Transform the MCG in accordance with the decision*. The decision of the previous step has effects on both time and power, again **pasap** and **palap** are used to maintain validity ie. ensure the $V1$ graph only

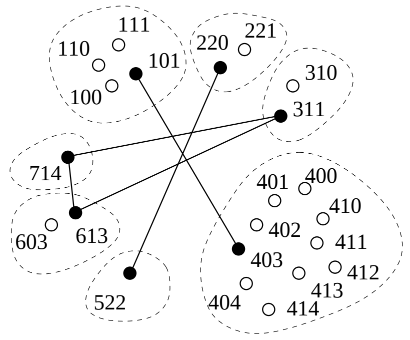


Figure 5. Partial-Clique partitioning. Shown are a set of V1 vertices, grouped (by the dotted lines) in operators. The only edges shown are those which are in the maximal clique not violating the power constraint .

contains the set of allowed vertices and allowed edges reflecting the current situation. Furthermore we need to preserve the cliques and disconnect those which no longer form one, refer to [3] for a detailed description.

Step 3: *Ensuring feasibility*. As **pasap** and **palap** are heuristic algorithms they depend on what operators have been scheduled, therefore a sequence of assignments might cause the of deletion unscheduled operators, causing an invalid schedule. The solution is to backtrack one step and lock the start time of all unscheduled operators to the pasap schedule (which was valid) and then continue, reducing our algorithm to a pure assignment and allocation algorithm from that point on.

Step 4: If any vertices left in $V1$, goto **Step 1**.

A comment to step 3, in most cases step 3 will not take effect and the algorithm will continue to the end, however it is possible to construct CDFGs which together with specific constraints causes the algorithm to execute this step. But even if it does, the algorithm has been allowed to operate for some time, during which it has significantly reduced area in comparison with the starting **pasap** schedule.

In figure 6 we illustrate the construction of a power-constrained schedule using our algorithm and the example CDFG. We will use the same time constraint T=5 and power constraint $P_< = 3$ as in figure 3. The onset of the algorithm is the construction of the **pasap** and **palap** schedules, shown in figure 3 and requiring at least 5 cycles for our power constraint, which generates the scheduling intervals for our operators. Using the scheduling intervals and our FU-library, shown in table 1, we generate the $V1$ graph shown in figure 6. Initially the algorithms creates a super-vertex of the

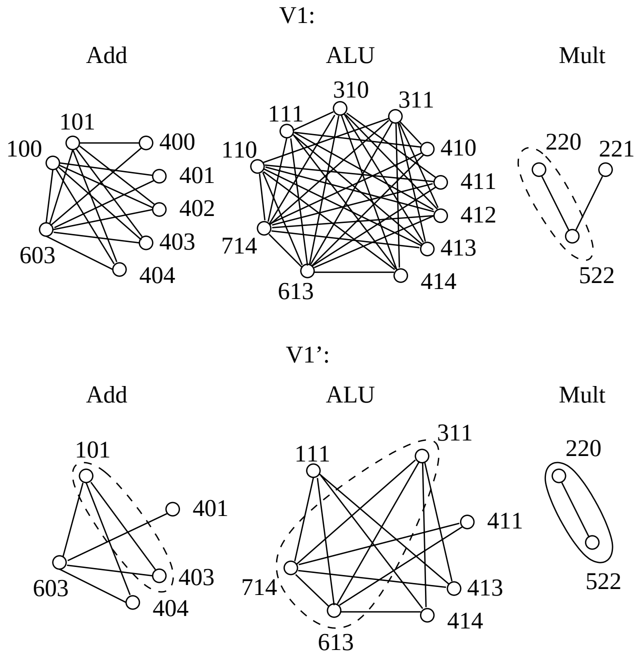


Figure 6. CDFG and the construction of the power constrained solution (T=5, P$_<$=3).

multiplier operation v_2 scheduled on Mult in cycle 0, then it merges v_5 scheduled on Mult in cycle 2 in to it, these are shown enclosed in the dotted ellipse.

The selection of v_2 scheduled on Mult in cycle 0 has consequences in the form of the **pasap** and **palap** algorithms deleting the following operations $\{100, 400, 110, 310, 410\}$ to maintain the V1 graph in a feasible state. Operation $\{221\}$ is deleted as v_2 now has been scheduled. Merging v_5 scheduled on Mult in cycle 2, similarly removes operations $\{402, 412\}$ and we arrive at the $V1'$ graph shown in figure 6 , with the super-vertex enclosed in the solid ellipse.

As it turns out the $V1'$ graph no-longer contains vertices (ie. cliques) which together with the super-vertices can violate the power constraint. Meaning the subsequent execution of the **pasap** and **palap** algorithms in principle reduces to execution of the **asap** and **alap** algorithms ie. the remaining part of the algorithms executes as the original algorithm in [3].

The final schedule, allocation and assignment corresponding is shown in figure 7, requiring one Add, one ALU and one Mult, using a total area cost of 6.5 units. Alongside the schedule is shown the power consumption in each cycle, where we now no-longer have a power violation as well as less spikes. We notice the price for the power constrained schedule compared with the non-power constrained schedule (using the same time-frame) is an extra adder, a relative

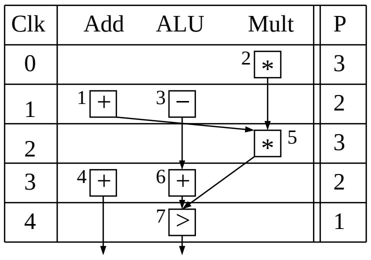


Figure 7. Solution (T=5,$P_<$=3) using one Add, one ALU and one Mult, using a total area of 6.5 .

Module	Oprs	Area	Clk-cyc.	P
add	$\{+\}$	87	1	2.5
sub	$\{-\}$	87	1	2.5
comp	$\{>\}$	8	1	2.5
ALU	$\{+,-,>\}$	97	1	2.5
Mult (ser.)	$\{*\}$	103	4	2.7
Mult (par.)	$\{*\}$	339	2	8.1
input	imp	16	1	0.2
output	xpt	16	1	1.7

Table 2. Functional unit library available to the synthesis algorithm.

area increase of 18 percent.

4. Results

We have bench-marked the algorithm on a set of CDFGs. We have done two sets of tests, all performed on a 200MHz Pentium II, with 96 MB memory:

pasap The first test is of the **pasap** algorithm where we investigate the required time delay of the CDFGs, as a function of the power constraint. The results are shown in table 3.

area vs. power The second test is of our main algorithm where we investigate the area of the resulting circuits as a function of the power constraint, with a constant time frame. We perform this test for a few different time-frames. The results are shown in table 5.

The FU library used in the tests is shown in table 2, the numbers does not represent an actual silicon library, but should be considered representative and illustrating the various trade-offs.

hal, vertices=21, edges=25							
$P_<$	inf	35	30	20	17.5	12.5	10	9
T_{pasap}	8	8	9	10	15	15	17	17

cosine1, vertices=57, edges=77								
$P_<$	inf	115	90	60	30	20	15	9
T_{pasap}	10	10	10	12	16	22	37	44
T_{palap}	10	10	12	14	18	22	38	44

Table 3. Time vs. power for the hal (top) and cosine (bottom) CDFGs.

CDFG	vertices	edges	$P_<$	T	A
fir7	24	24	inf	9	1852
			32	12	1116
			16	22	761
seventh	45	46	inf	14	2261
			32	22	1922
			20	27	1583
			16	41	1278
elliptic	49	43	inf	20	2516
			32	21	2110
			20	23	2182
			16	23	2023

Table 4. Different power and time constraints and the resulting area.

5. Discussion

As shown in table 5 (eg. hal with T=10 and cosine with T=12) using a global synthesis algorithm we can trade in area to obtain a solution which fits our power requirements. The average area penalty ranges in the region of 20 percent which is an acceptable penalty as today power has become the critical parameter.

We can classify the solutions found into 3 groups, as illustrated on figure 8:

(A) A hard time-constrained region, where time and the structure of the CDFG dictates the solution. Here there is little room for power constraints.

(B) A hard power-constrained region, where power and the structure of the CDFG dictates the solution. Corresponds to hal with T=17 and cosine with T=15,T=19.

(C) This is the region where we can trade-off power vs. time vs. area. Corresponds to hal with T=10 and cosine with T=12.

hal	$P_<$	inf	30	25	20	
T=10	Area	1109	1109	1109	1313	
hal	$P_<$	inf	30	15	10	9
T=17	Area	1016	1016	984	619	619
cosine	$P_<$	inf	150	100	80	70
T=12	Area	3158	3158	3164	3842	3826
cosine	$P_<$	inf	115	100	80	60
T=15	Area	3158	3164	3138	3138	3051
cosine	$P_<$	inf	70	40	35	30
T=19	Area	2306	2563	2306	2322	2645

Table 5. Power vs. area under different time constraints for our CDFGs.

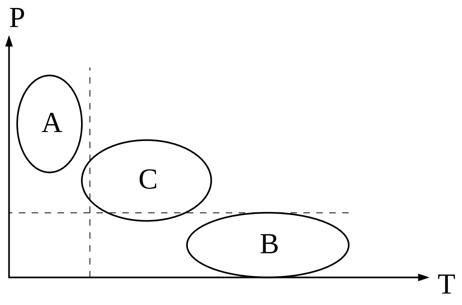


Figure 8. Solutions in the power-time space.

An interesting aspect is that with a large time and power constraint, the algorithm might find a worse solution with respect to area, than when the power constraint is tight. The reason for this lies in the greedy approach which might make a bad decision early on. With the tight power constraint this is prevented (no need to allocate many FUs in parallel if only one or two is used at a time).

Future work involves research into improving the power heuristic scheduling algorithms, **pasap** and **palap**, as well as taking into account the low-level power effects of assigning operators to FUs ie. minimizing internal signal activity.

6. Conclusion

In this paper we have presented an algorithm for time and power constrained synthesis of digital circuits. We have applied the algorithm on several examples and investigated different regions in the time-power-constraint space. The algorithm is capable of finding near optimal solution fulfilling the constraints and for the chosen silicon library we find the power constraint adds a modest increase in silicon area of roughly 20 percent. Furthermore the algorithm have an inherent tendency to “smoothen-out” the power schedule, a

$P_< = 17$	*hal*				
Clk	Operations				P
0	Imp_0	Imp_1	Imp_4	Imp_5	0.8
1	Imp_2	$Mult_6$	$Mult_7$		16.4
2	Imp_3				16.4
3	$Mult_8$	$Mult_9$			16.2
4					16.2
5	$Mult_{11}$	Add_{12}			10.6
6	Add_{10}				10.6
7	$Mult_{13}$	Sub_{15}	Exp_{20}		12.3
8					8.1
9	Les_{14}	Sub_{16}			5.0
10	Exp_{17}	Exp_{18}	Exp_{19}		5.1

Table 6. Tightly constrained power-schedule for the hal computation, T=11. Requiring 2 ALUs, 2 Mult(par), 4 Input, 3 Output, with a total area of $984\mu m^2$.

property which is highly desired.

References

[1] J. Luo and N. K. Jha. Battery-Aware Static Scheduling for Distributed Real-time Embedded Systems. In proceedings of 38th Design Automation Conference, 2001. Page(s): 444 -449.

[2] K.Lahiri, A. Raghunathan, S. Dey. Communication Architecture Based Power Management for Battery Efficient System Design. In proceedings of Design, Automation and Test in Europe. 2002. Page(s): 691 -696

[3] J. Jou; S. Kuang; R. Chen. Clique Partitioning Based Integrated Architecture Synthesis for VLSI Chips. In proceedings of International Symposium on VLSI Technology, Systems, and Applications, 1993, Page(s): 58 -62.

[4] Y. Choi and T. Kim. An Efficient Low-Power Binding Algorithm in High-Level Synthesis. Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 4 , 2002 Page(s): 321 -324.

[5] M. S. Bright and T. Arslan. Synthesis of Low-Power DSP Systems Using a Genetic Algorithm. In IEEE Transactions on Evolutionary Computation, vol. 5, no. 1, Feb 2001.Page(s): 27 -40.

[6] F. Gruian and K. Kuchcinski. Operator Binding and Scheduling for Low Power Using Constraint Logic Programming. Euromicro Conference, 1998. Proceedings. 24th , Volume: 1 , 1998 Page(s): 83 -90 vol.1.

[7] M. Bhardwaj, R. Min, A. Chandrakasan. Power-Aware Systems. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , Volume: 9 Issue: 6 , Dec. 2001 Page(s): 757 -772.

[8] J. Liu, P. H. Chou, N. Bagherzadeh, F. Kurdahi. Power-Aware Scheduling under Timing Constraint for Mission-Critical Embedded Systems. In proceedings of the 38th Design Automation Conference, 2001. Page(s): 840 -845.