TOWARDS LOW-POWER SYNTHESIS: A COMMON SUB-EXPRESSION EXTRACTION ALGORITHM UNDER DELAY CONSTRAINTS

Ihab Amer and Wael Badawy
Department of ECE
University of Calgary
Calgary, Alberta, Canada, T2N 1N4
{amer, badawy}@enel.ucalgary.ca

Muhammed Mudawwar
Computer Science Department
The American University in Cairo
113 Kasr El Aini Street, Cairo, Egypt
mudawwar@aucegypt.edu

ABSTRACT

This paper presents a common sub-expression extraction algorithm targeting the reduction of power consumption of multi-level combinational logic networks under delay constraints. The proposed algorithm has been prototyped and simulated using ORCAD 9.2®. The results show that adding delay constraints prevents the combinational logic network from suffering uncontrollable degradation in performance during the optimization process for reducing power consumption.

I. INTRODUCTION

With the expansion of various mobile devices, power minimization of digital systems becomes greatly required. In fact, the absence of low-power design techniques may lead to the situation that future portable devices will suffer from either a very short battery life or a very heavy battery pack [1]-[2]. In addition to portable devices, high-end products strongly require reduction in power consumption due to the huge cost associated with various packaging and cooling strategies [3]-[4]. Moreover, high cost of power consumption and some environmental demands made it very essential to search for various techniques to reduce power consumption [5]-[6].

High-level synthesis is a design technique that can optimize power consumption at different levels of abstraction [7]. *Common sub-expression extraction* targeting low power is a popular method that was proposed by Iman and Pedram in [8]. It is an algebraic optimization technique that can be applied to a set of logic functions in a Boolean network to determine an "optimal" set of logic functions such that when this set is inserted into the network, the "cost" of the network is minimized [9]. The main limitation of the above algorithm is the difficulty of applying it to systems that are required to work under delay constraints. This is because the algorithm may result in uncontrollable degradation in performance of the logic network that is being optimized.

This paper presents an enhancement of the algorithm proposed by Iman and Pedram. It provides it with the ability to work under delay constraints and hence it prevents the optimized logic network from suffering severe degradation in performance.

The rest of this paper is organized as follows: Section II reviews the algorithm that was proposed by Iman and Pedram. In section III, the proposed enhancement of the algorithm is introduced. Section IV presents the prototype and performance analysis of the proposed algorithm. Finally, section V concludes this paper.

II. COMMON SUB-EXPRESSION EXTRACTION FOR LOW POWER

Common sub-expression extraction is an algebraic technique that can be used to optimize multi-level combinational logic networks. The extraction process relies on the search of common divisors of two (or more) expressions. The "best" one (if any) is chosen to be extracted representing a new local function of the network, and its corresponding variable can be used to simplify the original expression [10].

Figure 1 shows a flowchart for the overall optimization process targeting low power.

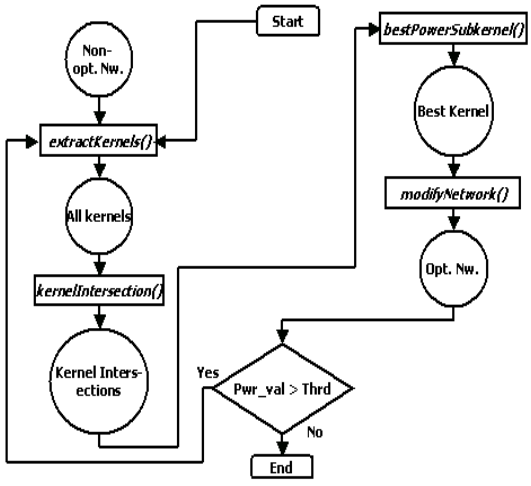

Figure 1. A flowchart for the overall optimization process targeting low power

The input is a non-optimized combinational logic network while the output is a functionally equivalent network with an altered topology and with minimum

power consumption that can be achieved using the algorithm. This is done by repeatedly extracting *best* sub-expressions (usually called *kernels*) until the algorithm senses that more computations might be non-significant from the designer's point of view.

The procedure *bestPowerSubkernel()*, presented in Figure 2 shows that the algorithm searches greedily for the kernel with the highest *power_value* (i.e. the kernel whose extraction leads to the largest power reduction) disregarding any degradation that may occur in performance. This appears in the following pseudo-code of the procedure *bestPowerSubkernel()*.

```
bestPowerSubkernel(){
bestK = K1; //Initialize bestK with the
            //first kernel in the kernel-
            //intersection list
for i = 2 to k{      //where k is the number of kernel
                     //intersections
    power_value = calcPower(Ki);
    if(power_value > power_value  of bestK)
                     bestK = Ki;
   }
return(bestK);
}
```

Figure 2. Pseudo-code of the Iman/Pedram *bestPowerSubkernel()* procedure

III. THE PROPOSED ENHANCEMENT

The proposed algorithm handles the limitation in the procedure *bestPowerSubkernel()* shown in Figure 2. Each time a decision needs to be taken to choose the kernel to be extracted, it checks that the delay constraints are satisfied rather than returning the kernel whose extraction gives the highest reduction in power consumption disregarding the effect of this extraction on the performance of the logic network.

It is important to keep the optimization process work under delay constraints due to the fact that the output optimized combinational logic circuit is most probably going to be placed as a control circuit inside a whole sequential circuit. Thus, an important design issue is that the control circuit should not represent a bottleneck in the whole design.

Delay constraints can be represented in several forms. The simplest is by providing the optimization process with the maximum number of logic levels that are allowed to exist in the critical paths of all the logic functions in the network.

Our main contribution is to change the procedure *bestPowerSubkernel()* so that it returns the kernel whose extraction results in the highest reduction in power consumption, however it still satisfies the timing constraints that are provided by the user (the digital designer in our case). This can be shown in the following pseudo-code.

```
bestPowerSubkernel(){
sort(a pointer to the set of all kernel intersections);
for i = 1 to k{       //where k is the number of kernel
                      //intersections
        if(satisfyTimeConstraints(Ki)){
                      bestK = Ki;
                      break;
                      }
}
        return(bestK);
}
```

Figure 3. Pseudo-code of the proposed *bestPowerSubkernel()* procedure

Thus, all the kernel intersections are initially sorted in descending order according to their *power_value*. Then, they are checked for satisfying delay constraints until a kernel is found whose extraction does not break the timing constraints. This kernel is the best one to be extracted, hence it is returned by the procedure.

The procedure *satisfyTimeConstraints()* returns TRUE if the extraction of the examined kernel does not break the delay constraints. The procedure checks that the actual data-ready time of each of the function expressions in the network does not exceed the output required data-ready time after the extraction operation. This is done by repeatedly calling the procedure *checkFunction()* for each of the functions in the logic network as shown in the following pseudo-code.

```
satisfyTimeConstraints(kernel k){
//k is the kernel being examined
        satisfied = TRUE;
        for i=1 to m{  //where m is the number of
                       //outputs of the network
            if(checkFunction(fi, k) == FALSE){
                satisfied = FALSE;
                break;
             }
        }
        return(satisfied);
 }
```

Figure 4. Pseudo-code of the proposed *satisfyTimeConstraints()* procedure

The procedure *checkFunction()* is responsible for checking whether the extraction of the examined kernel will result in breaking the delay constraints of a certain function or not. The contents of the procedure *checkFunction()* differs according to the model used for estimating delay.

IV. PROTOTYPING AND PERFORMANCE ANALYSIS

A very demanding issue is to test the results that are reported by the algorithm after enhancement. Hence, the choice of the experimentation scheme is very important.

We spot on what we might consider simple logic networks. However, we study these networks thoroughly, considering the three different topologies that might be taken by the logic network (the original non-optimized network, the network after optimization for low power without delay constraints, and the network after optimization for low power under delay constraints).

The choice of "simple" logic networks stems from the fact that the probability of appearance of glitches in a certain node of a logic network increases with the increase in the size of the network. Thus, the choice to verify the developed algorithm (which is designed for a zero-delay model) using simple logic network is to decrease the possibility of appearance of glitches in different nodes of the network. This guarantees that the simulated power consumption almost results from useful transitions only. Besides, if the algorithm works well with "simple" logic networks, then it is expected that it is going to work better with "complex" networks where there is a great pool of common kernels to be extracted.

A logic-level CAD tool is needed to handle the verification process. ORCAD 9.2® of Cadence™ is used for this purpose.

Two examples of multiple-output combinational logic networks are chosen to be our test benches during experimentations: A two-bit comparator and a two-bit adder.

The obtained results show that after implementing the suggested modifications, the performance of the logic network is prevented from being severely damaged, which in turns enables the designer to include the network in a sequential circuit operating under higher speed clocks.

The following table summarizes the results that are obtained during the experimentation phase.

Test Bench	Volt	Bit Rate	Original Network				Network Optimized for Low Power Only				Network Optimized for Low Power Under Delay Constraints			
			No. of gates	Power Dissp.	Max Delay	Max Opr. Freq.	No. of gates	Power Dissp.	Max Delay	Max Opr. Freq.	No. of gates	Power Dissp.	Max Delay	Max Opr. Freq.
2-bit Comp.	5V	500 Kbps	27	540 units	462 units	2.16 units	21	210 units	590 units	1.69 units	19	385.6 units	335 units	2.98 units
2-bit Adder	5V	500 Kbps	35	1147.5 units	462 units	2.16 units	25	1488.5 units	844 units	1.18 units	25	880 units	450 units	2.2 units

Table 1. Summary of obtained results

The table shows that for the two-bit comparator, when the logic network is optimized for low power without delay constraints, a 61.1% reduction in power consumption is achieved. However, when the optimization is performed under delay constraints, a 28.6% reduction in power consumption is achieved, but the maximum operating frequency of the logic network becomes about 38% better than its corresponding value when the optimization process is performed without delay constraints. Therefore, our proposed technique is preferable when a reduction in power consumption of the system is required, but in the same time, the minimum value of the clock that is going to synchronize the system is an important design issue.

Some of the results in the table may appear unexpected. For instance, for the two-bit adder, the network optimized for low power only consumes more power than the network optimized for low power under delay constraints. This is due to the appearance of glitches, which increase the power consumption severely.

V. CONCLUSION

In the present paper we have developed an optimization tool that accepts a non-optimized combinational logic network as an input and produces a network with reduced power consumption, yet under delay constraints.

The system has been tested using ORCAD 9.2®. Two examples of multiple-output combinational logic networks were chosen to be our test benches during our experiments: A two-bit comparator and a two-bit adder.

REFERENCES

[1] M. Pedram, "Power Minimization in IC Design: Principles and Applications," *ACM Transactions on Design Automation of Electronic Systems,* Vol. 1, No. 1, pp. 3-56, January 1996.

[2] P. Havinga, "Mobile Multimedia Systems," *Ph.D. thesis, University of Twente*, February 2000.

[3] P. Gray and R. Meyer, "Analysis and Design of Analog Integrated Circuits. Third Edition," *John Wiley & Sons, Inc. Canada*, 1993.

[4] C. Small, "Shrinking devices put the squeeze on system packaging," *EDN 39*, *4*, pp. 41–46, February 1994.

[5] T. Mudge, "Power: A First-Class Architectural Design Constrain," *IEEE J. Computer*, *IEEE Press*, pp. 52-58, 2001.

[6] G. Cai and C. Lim, "Architectural Level Power/Performance Optimization and Dynamic Power Estimation," *Cool Chips Tutorial: An Industrial Perspective on Low-Power Processor Design, T. Mudge, S. Manne, and D. Grunwald, eds., IEEE CS Press, Los Alamitos, Calif.*, pp. 90-113, 1999.

[7] L. Benini, and G. De Micheli, "System-Level Power Optimization: Techniques and Tools," *ACM Transactions on Design Automation of Electronic Systems,* Vol. 5, No. 2, pp. 115-192, April 2000.

[8] S. Iman and M. Pedram, "Logic Synthesis for Low Power VLSI Design," *Kluwer Academic Publishers. USA*, 1998.

[9] J. Rabaey and M. Pedram, "Low Power Design Methodologies. Fourth Printing," *Kluwer Academic Publishers USA*, 2001.

[10] G. De Micheli, “Synthesis and Optimization of Digital Circuits,” *McGraw Hill. USA*, 1998.