Reprint from *Proceedings of the International Conference on Computer-Aided Design*, 1996

Exploiting Regularity for Low-Power Design

Renu Mehra and Jan Rabaey

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720
email: {renu, jan}@eecs.berkeley.edu

Abstract — *Current day behavioral-synthesis techniques produce architectures that are power-inefficient in the interconnect. Experiments have demonstrated that in synthesized designs, about 10 to 40% of the total power may be dissipated in buses, multiplexors, and drivers. We present a novel approach targeted at the reduction of power dissipation in interconnect elements — buses, multiplexors, and buffers. The scheduling, assignment, and allocation techniques presented in this paper exploit the regularity and common computational patterns in the algorithm to reduce the fan-outs and fan-ins of the interconnect wires, resulting in reduced bus capacitances and a simplified interconnect structure. Average power savings of 47% and 49% in buses and multiplexors, respectively, are demonstrated on a set of benchmark examples.*

1. Introduction

In recent years, low power has become a primary design concern. Among the different power consuming components of a chip, the interconnect components — buses, multiplexors, and buffers — are the focus of this work. The importance of targeting interconnect power reduction at the architecture level is highlighted by the following two facts — (i) interconnect components may consume a large percentage of the total power and (ii) their power consumption is highly dependent on architecture-level design decisions [1].

We provide an scheduling, assignment, and allocation strategy specifically aimed at reducing interconnect power. We target ASIC implementations of datapath-intensive, real-time DSP applications with fixed throughput constraints. The main idea behind our approach is to exploit the regularity inherent in the algorithm to derive a simplified interconnect structure in the final implementation.

1.1 The impact of exploiting regularity

Regularity in an algorithm refers to the repeated occurrence of computational patterns, e.g., multiply-add patterns in an FIR filter and bi-quads in a cascade-form IIR filter. We exploit the regularity of an algorithm by detecting repetitive patterns in it and mapping them such that corresponding nodes in different instances of the pattern are mapped to the same hardware unit. As a result, connections within a pattern are instantiated only once and are *reused* in each instance of the pattern. This leads to a simplified interconnect structure with reduced fan-ins and fan-outs.

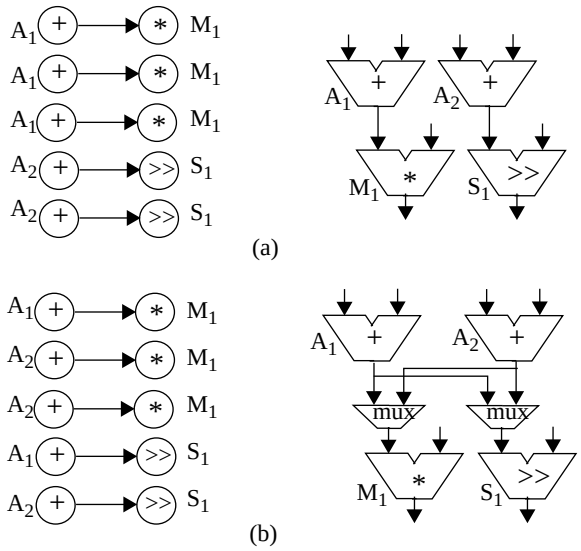


Figure 1. Preserving regularity leads to a simplified interconnect structure, (a) Regular assignment, (b)Non-regular assignment.

Fig. 1 shows two different assignments for a part of an algorithm and the corresponding hardware netlists. In the first case, all the instances of the add-mult pattern are assigned to the same adder-multiplier pair (A_1 and M_1). As a result, the connection from the adder to the multiplier needs to be instantiated only once and can be reused without any multiplexing overhead. The assignment of Fig. 1(b) does not preserve regularity. Here the output of the adder A_1 connects to both M_1 and S_1 and requires more multiplexing. This example shows that a regular assignment leads to less fan-outs and fan-ins and lower multiplexing overhead.

Power reduction in a regular implementation stems from two factors. Due to reduced fan-outs, the interconnect lines can be kept short leading to lower switched capacitance. These reduced fan-out buses are used often (for data transfers in recurring patterns) giving the desirable combination of reduced capacitance on the more active buses. Secondly, since the fan-outs and fan-ins of hardware units are reduced,

ICCAD '96
1063-6757/96 $5.00 © 1996 IEEE

the multiplexing overhead in terms of the buffers and multiplexors required is decreased.

Note that, since a regular implementation is more constrained than a non-regular one, it may require more hardware units and the power savings may come at the cost of increased area.

1.2 Related work

Originally, most high-level systems focused on functional unit optimizations. Recently, as research showed that the interconnect has a first order effect on the quality of the overall design [2], there has been a growing interest in interconnect optimization. Several high-level synthesis systems have incorporated interconnect minimization as one of the primary goals [3, 4]. However, none of these have targeted power reduction — they reduce the number of buses but ignore the cost of accessing them. Techniques for interconnect power optimization by exploiting the locality of the algorithm are presented in [1]. The approach in that work is complementary to, and can potentially be used in combination with, the current approach.

Techniques to preserve and exploit regularity have been gaining interest because many algorithms have repeated computational patterns, especially in the DSP domain where a large set of component applications — FIR and IIR filters, Fourier and cosine transforms, etc., inherently have a high degree of regularity. In high-level synthesis, the regularity issue has been addressed for both speed and area before [4-8]. However, no work has been done to exploit regularity for low power.

2. Overall approach

This section explains our overall approach. We first present the targeted architecture model and relevant terminology.

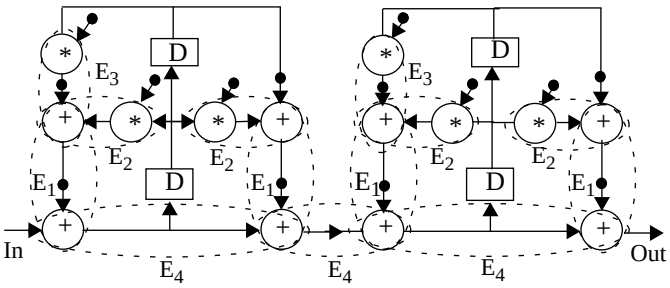
2.1 Architecture model

We target the following architecture model. Each functional unit has dedicated single-ported register files at its inputs to store the variables it needs. A variable is written into the register file when its producer operation is executed. The interconnect structure is multiplexor-based with no tri-state buffers. Under this model, each functional unit has a dedicated output bus which can fan out to one or more destinations. Multiplexors are used at the inputs of the units to select the appropriate input bus in different time-steps.

2.2 Terminology

The algorithm is represented as a data-flow graph where nodes represent algorithm operations and edges represent data transfers. We define an *E-instance* as a pair of nodes connected by an edge, so named since it is derived from an edge of the graph. E-instances are classified into types, or *E-templates*, based on the type of their input and output ports. The *coverage* of an E-template is defined as the number of E-instances of that type divided by the total number of edges in the graph. It represents the degree of recurrence of the E-template.

E-templates of a fourth-order cascade filter and the corresponding coverages are shown in Fig. 2 (edges to the right input ports are indicated with a dot). For example, the E-template E1, from an add operation to the right input of an add operation, occurs four times. Currently our implementation does not allow permutations of inputs to commutative operations which would enable further exploration of the design space and improve the results.



E-template	Coverage
E1 (add → add.right)	4/26
E2 (mult → add.left)	4/26
E3 (mult → add.right)	2/26
E4 (add → add.left)	3/26

Figure 2. Some E-templates in a fourth-order cascade filter.

2.3 Using E-templates in synthesis

The main tasks in architecture synthesis are scheduling, assignment, and allocation. For a given clock speed and algorithm throughput, the *scheduling* process assigns each operation in the data-flow graph to one or more time steps. The goal is to minimize the total area while scheduling all the operations within a given performance constraint. In our approach, along with the cost of each hardware, a cost is assigned to each E-template representing the cost of the connection between the input and output nodes. The new scheduling algorithm minimizes the cost of the E-templates along with the overall area. The aim here is to derive a schedule that *enables* a regular assignment of operations to hardware.

The *assignment* task binds operations in the algorithm to specific hardware units and the *allocation* task decides the number of resources of each type to be used. In our methodology, the scheduling is performed first and then allocation and assignment are done simultaneously. *The main idea*

behind our assignment-allocation scheme is to assign E-templates as a whole in order to preserve the two-node regularity of the algorithm. Thus the data transfers of E-instances assigned to the same pair of hardware units can use the same bus without any extra multiplexors or buffers, and without increasing the fan-out of the bus. Consider the E-template E2 of the cascade filter shown in Fig. 2. If, instead of assigning individual nodes, we assign the corresponding E-instances of this template to a multiplier-adder pair, we ensure that the output of the multiplier goes only to the left input of the corresponding adder. Fan-outs of the buses from each multiplier is kept low and each of these buses once instantiated can be reused for the four data transfers without any multiplexing overhead. A similar idea to reduce interconnections during assignment for pipelined datapaths is given in [4] where the authors consider assignment of paths (not E-templates) and propose a technique different from ours.

Using E-templates as opposed to larger patterns for exploiting regularity has the advantage that, while detecting and matching generic patterns is NP-complete, these operations take linear time for E-templates. Our results indicate that large power savings can be achieved with the E-template based approach. Sections 3 and 4 present the details of our scheduling and assignment-allocation techniques based on this approach.

3. E-template-based scheduling

Our scheduling approach derives from the force-directed scheduling technique first proposed by Paulin [9]. For a detailed description of the algorithm we refer readers to that paper, here we limit the discussion to its effect on assignment regularity. Consider an example with two E-templates, E1 and E2, with four and two instances, respectively, as shown in Fig. 3(a). From the ASAP and ALAP times (marked next to each node), it is clear that it is possible to map the multiply operations of all multiply-add E-instances (E1) to the same multiplier and similarly those of the multiply-shift E-instances (E2). The initial distribution graph for multiplications (refered to in this work as *functional-unit distribution-graph* or *FDG*) and their schedule obtained using the force-directed algorithm are shown in Figs. 3(b) and 3(c), respectively. In this schedule the height of the DG is minimized, but it is not possible to map the multiply operations of the all multiply-add E-instances to the same unit since *b* and *c* are scheduled in the same time-step.

As shown in this example, a force-directed schedule may preclude the preserving of regularity in a graph since it does not consider the cost of connections. We propose a modification that accounts for the cost of the connections and represents them as *connection distribution-graphs* (*CDGs).*

Each E-template has two CDGs — one for its sources and one for its destinations. These distribution graphs represent the cost of the interconnect between the source and destination nodes. For a given E-template, E_k, the CDG for sources is derived from the time distributions of the source nodes of all instances of the E-template, while the CDG for destinations is derived from distributions of the destination nodes.

The total force on any node is the weighted sum of the forces from the FDG of the relevant functional unit, the source CDGs of all the E-templates for which this node is the source node and the destination CDGs of all the E-templates for which this node is the destination node. The weight of an FDG is proportional to the cost of the unit while the weight of each CDG is proportional to the coverage of the corresponding E-template. This weighting scheme gives preference to connections that are repeated more often.

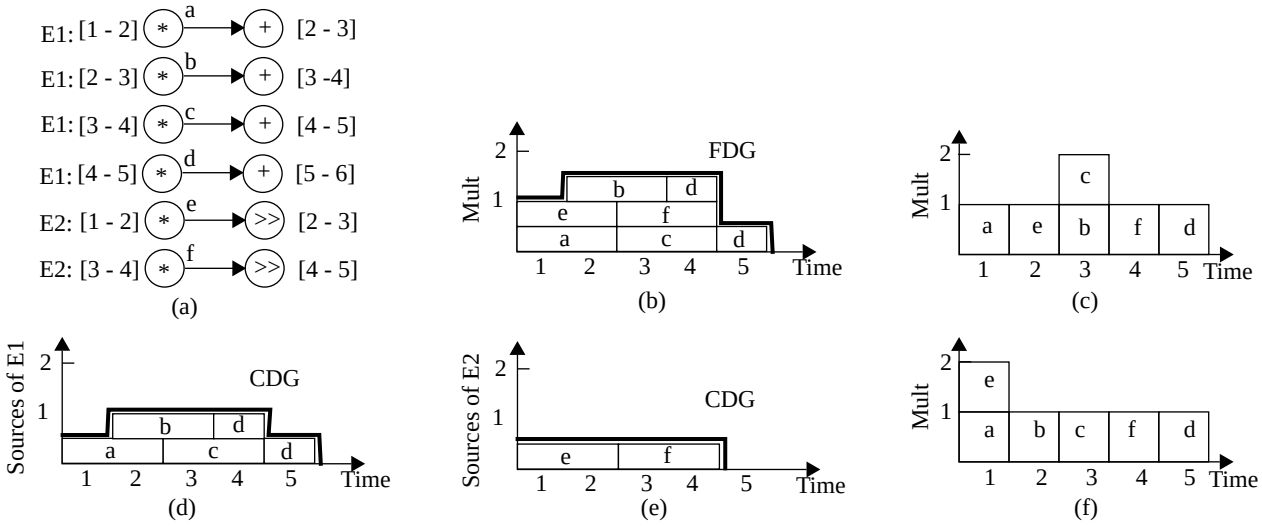


Figure 3. The effect of using connection distribution graphs, (a) Instances of two E-templates with their ASAP and ALAP times, (b) Initial FDG for multiply operations, (c) Final distribution graph using only FDGs, (d, e) Initial source CDGs of the two E-templates, (f) Final distribution graphs using FDGs and CDGs.

This modified force-directed scheduling approach attempts to aid the assigning of E-instances of the same E-template to the same pair of hardware units while also minimizing the total area.

Consider the example of Fig. 3 again. The initial source-CDGs of the two E-templates are shown in Fig. 3(d, e) and the final distribution graph that minimizes the weighted sum of the FDG and CDGs are shown in Fig. 3(f). Notice that multiply operations of all multiply-add E-instances are scheduled at different time slots since the scheduler minimizes the height of its source-CDG along with that of the FDG, and therefore, they can be mapped onto the same hardware unit. Similarly the multiply operations of all multiply-shift E-instances can be mapped to the same multiplier.

4. E-template based assignment and allocation

Our assignment and allocation strategy strongly hinges on the concept of a conflict graph and its maximum independent set which we first explain in Sections 4.1 and 4.2, respectively. In Section 4.3, we describe the overall assignment and allocation algorithm.

4.1 Conflict graphs

The conflict graph, C_k, for an E-template, E_k, is derived in the following way. Each unassigned E-instance (for which at least one node — source or destination — is unassigned) of type E_k is represented by a node in the conflict graph (*conflict-node*). Two conflict-nodes are joined by an edge if the sources or destinations of the corresponding E-instances cannot be assigned to the same hardware unit. This occurs if any of the following four conflicts exists between either the sources or destinations of the corresponding E-instances.

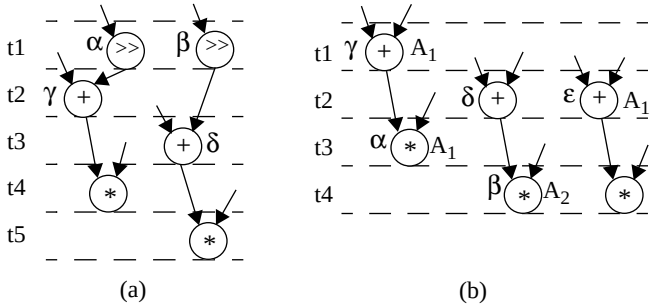


Figure 4. Types of conflicts, (a)Scheduling and register bandwidth conflicts, (b)Assignment and assign-schedule conflicts.

Scheduling conflict — A scheduling conflict exists between two nodes if there is an overlap in the time slots in which they are scheduled (e.g., between nodes α and β in Fig. 4(a)).

Register-bandwidth conflict — Due to the distributed, single-ported nature of register files in our hardware model (Section 2.1), there is a register-bandwidth conflict between two nodes if the producers of their corresponding inputs are scheduled in the same time slot. In Fig. 4(a), there is a register bandwidth conflict between nodes γ and δ since node α writes into the right port of γ at the same time at β writes into the right port of node δ.

Assignment conflict — An assignment conflict is introduced if the nodes are assigned to different functional units. (e.g., between nodes α and β in Fig. 4(b) since they are assigned to different adders, A_1 and A_2).

Assign-schedule conflict — An assign-schedule conflict is introduced if the one of the nodes is already assigned to a hardware resource and the other has a scheduling or register-bandwidth conflict with that hardware resource. A node is said to have a scheduling or register-bandwidth conflict with a hardware resource if it has a scheduling or register bandwidth conflict, respectively, with any of the nodes that are assigned to that resource. In Fig. 4(b), there is a assign-schedule conflict between nodes γ and δ since δ has a scheduling conflict with A_1, the hardware resource that α is assigned to.

4.2 Maximum independent set

The maximum independent set (MIS) of a graph is defined as the largest subset of nodes of the graph, such that there does not exist an edge between any pair of nodes in that subset. The MIS of the conflict graph, is the maximum set of E-instances with no conflict edges between them and therefore represents the largest set of E-instances that can be assigned to the same pair of hardware units.

We derive the maximum independent set using a popular greedy heuristic that has been shown to give good results [10]. This algorithm is modified to bias it towards choosing the more favorable candidate in case of a tie.

4.3 E-template based assignment strategy

Our assignment-allocation scheme is divided into two phases. We start by detecting all the E-templates in the given graph and calculating their coverages.

The first phase of the algorithm iteratively assigns sets of E-instances to pairs of hardware units. In each iteration, the E-template with the highest coverage (in case of a tie, the one with a higher MIS cardinality) is selected, the MIS of its conflict graph is calculated, and the corresponding E-instances are assigned. The sources of the E-instances are assigned first. If any of the source nodes are already assigned, all others are assigned to the same unit. Otherwise, a new hardware unit is allocated and assigned to all the source nodes. The destination nodes are then mapped in the same way. Assigned E-instances are removed from the E-template list and the coverage of the E-template is recalculated.

Notice that it is not possible for the source nodes (or the destination nodes) of a pair of E-instances in the MIS to be already assigned to different hardware units since this would have caused an assignment conflict between them. Also, if only one of them is assigned to a unit, the other node can also be assigned to the same unit since there is no assign-schedule conflict between them.

As more nodes get assigned, the number of E-instances mapped in each iteration reduces due to reduced coverages and increased assignment and assign-schedule conflicts. As a result the advantages from the reuse of the dedicated connections between the corresponding hardware units — reduced muxes and bus fan-outs — are decreased and the area overhead is increased, reducing the benefits from E-template based assignment. In each iteration, therefore, E-templates whose coverages fall below a certain threshold are eliminated.

When all the E-templates are eliminated, the first phase terminates and the remaining nodes are colored using a vertex coloring technique [11]. A pseudo code for the assignment and allocation algorithm is given in Fig. 5.

```
ETemplateList = MakeETemplates(OriginalGraph)
CalculateCoverage(ETemplateList)
RemoveETemplatesWithCoverageBelowThreshold(ETemplateList)
Best_ETemplate = SelectBestETemplate(ETemplatesList)
while (Best_ETemplate != NULL) {
        ConflictGraph = CreateConflictGraph(Best_E-Template->List
        MIS_List = MaxIndependentSet(ConflictGraph)
        Allocate_and_AssignList(MIS_List)
        UpdateETemplates(ETemplateList)
        CalculateCoverage(ETemplateList)
        RemoveETemplatesBelowThreshold(ETemplateList)
        BestTemplate = SelectBestETemplate(ETemplateList)
}
ResidualList = MakeListOfUnassignedNodes(OriginalGraph)
VertexColoring(ResidualList)
```

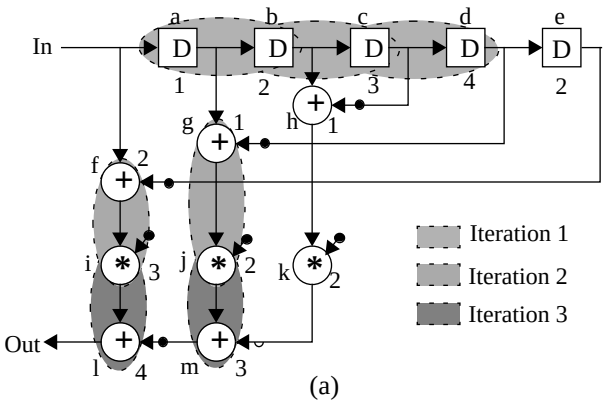
Figure 5. Pseudo-code of the assignment and allocation algo-

The algorithm greedily selects E-templates that are repeated very often since the aim is not to reduce the total number of fanouts but rather to reduce fanouts (and hence capacitance) of buses that are accessed often.

4.4 Example

In this section we demonstrate the operation of the algorithm on a small example. Consider the reverse symmetric FIR filter shown in Fig. 6(a). The numbers next to the nodes in show the time steps each node is scheduled in. Fig. 6(b) shows the E-templates and their coverages. The coverage threshold is set at 2/16.

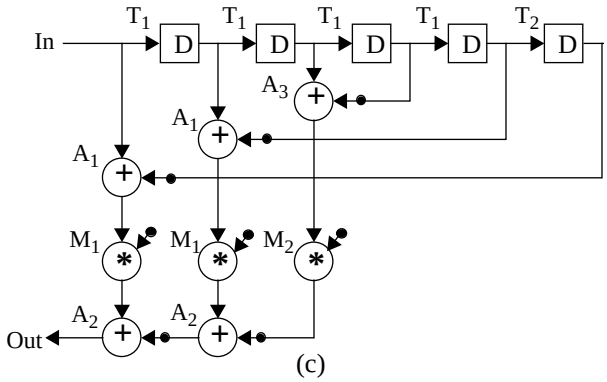
Phase1, Iteration 1 — E-template E_0 is selected for assignment. The selected MIS of E-instances is *a-b*, *b-c*, *c-d* (scheduling conflict between their destination nodes of *a-b* and *d-e*)[1]. A delay unit, T_1, is allocated and the source nodes, *a*, *b*, and *c* are assigned to it. As a result of this, some destination nodes get assigned to T_1 and therefore, the rest are also assigned to T_1. Since the coverage of E_0 falls below the threshold, it is removed from the E-template list.



(a)

E-template name	Description	Coverage
E_0	D → D	4/16
E_1	D → add.left	2/16
E_2	D → add.right	3/16
E_3	add → mult.left	3/16
E_4	mult → add.left	2/16

(b)



(c)

Figure 6. Effect of E-template based assignment on, fifth-order reverse-symmetric IIR filter, (a) E-templates assigned in each iteration, (b) E-templates and their coverages, (c) Final assignment.

Iteration 2 — E-templates E_2 (*c-h*, *d-g*, *e-f*) and E_3 (*f-i*, *g-j*, *h-k*) have the highest coverage and their MIS cardinalities are 1 (assignment conflicts between *c* & *e*, and *d* & *e*; and

[1]This is a special case not discussed here for brevity. Since the source and destination nodes are of the same type, two conflict graphs are generated — one that allows sharing between sources and destinations and one that does not — and the one with higher MIS cardinality is selected.

scheduling conflict between *g* & *h*) and 2 (scheduling conflict between *g* & *h*), respectively. E_3 is selected and sources and destinations of its MIS (*f-i, g-j)* are assigned to multiplier M_1 and adder A_1, respectively. The coverages of unassigned instances of E_1, E_2, and E_3 drop below the threshold and they are eliminated.

Iteration 3 — E-template E_4 is selected and both its instances are assigned to the multiplier, M_1, and adder, A_2, pair.

Since all E-templates are now eliminated, the remaining nodes are assigned using vertex coloring. The E-instances assigned in each iteration is shown in Fig. 6(a) and the final assignment obtained is shown in Fig. 6(c).

5. Interconnect models

The power savings in our synthesis strategy stem from the reduction of power consumed in buses and multiplexors and it is important to estimate the power consumed by these components in order to validate our synthesis strategy. We used *SPA*, an architectural power analysis tool [12], for our estimations. The power consumed by buses depends on the length of buses which is difficult to estimate before placement and routing. In order to analyze the effect of the synthesis technique on power, we first present a model for the estimating bus lengths. The model has been validated using layouts.

At the gate level, wire lengths are modeled as being directly proportional to the fan-out of the wire [13] but this effect is largely ignored in architecture-level models [14, 15]. Examining several designs we found that the linear relationship holds even at the high level. The length of any bus, *i*, is estimated as L_{pp} times its fan-out, F_i, as given in Equation 1. L_{pp} represents the length of a bus with single fan-out and is constant over all buses for a given design.

$$L_i = F_i L_{pp} \quad (1)$$

The length, L_{pp}, of the a bus with a single fan-out is assumed to be proportional to the square root of the area of the chip [14, 16].

$$L_{pp} = \gamma\sqrt{A_{chip}} \quad (2)$$

The chip area is found using the model presented in [14]. The constant in the model, γ, was found empirically from designs obtained from both the Hyper [17] and the E-template based synthesis systems. It was determined to be 0.72, 0.80, 0.81, 0.88, 0.80, and 0.68 for the six chip-layouts generated. The mean value of γ, 0.78, was selected for our model.

Besides the capacitance of the wire itself, the capacitive load on it is switched when the bus is accessed. We used a fixed capacitive load (50fF in our 1.2 micron technology) on each fan-out. The above models were implemented in *SPA*.

6. Results

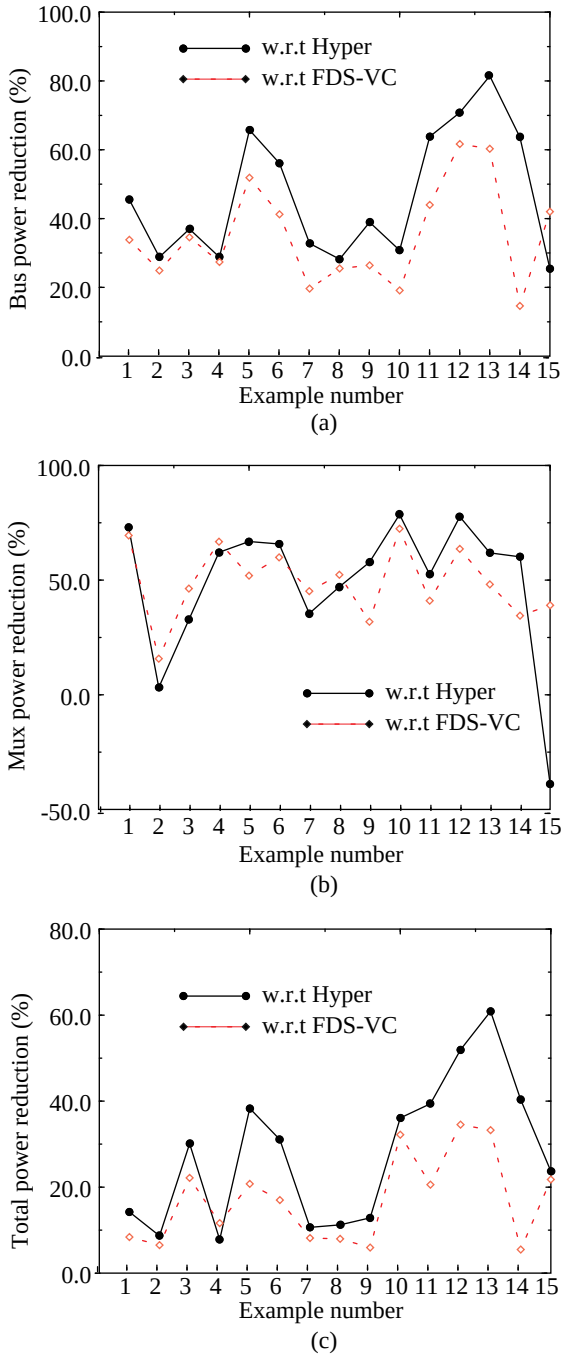


Figure 7. Percentage power savings with respect to the Hyper and FDS-VC schemes, (a) Buses, (b)Multiplexors, (c) Total.

This section compares the quality of results obtained from the E-template based synthesis methodology with two other scheduling/assignment paradigms — the Hyper synthesis scheme [17] and a force-directed scheduling followed by vertex-coloring assignment (FDS-VC). A set of 15 examples, consisting of different structures of FIR filters, IIR filters and transforms were selected for experimentation. All

algorithms were in their original forms (not transformed) and were evaluated for maximum throughput implementations (total time available equal to critical path). We used *SPA* with uniform white noise models to decouple the power savings due to regularity exploitation from those due to changes in signal correlations.

The graphs in Fig. 7 show the percentage improvements in bus, multiplexor, and total power compared with the Hyper and the FDS-VC implementations. As compared to Hyper, an average of 47% and 49% power savings were obtained for buses and multiplexors, respectively, while compared to FDS-VC, the average reductions in these components was 39% and 49%, respectively. Overall average power reductions of 28% and 17% were obtained with respect to the Hyper and FDS-VC synthesis schemes, respectively. We also expect to obtain power savings in buffers since smaller buffers can be used to drive the low fan-out, short buses. However, since our automated architecture-netlist generation tool uses minimum sized buffers for all data transfers, irrespective of the length of the bus being driven, we are not able to demonstrate these savings.

Fig. 8 shows the percentage change in the total chip area with respect to the Hyper and FDS-VC implementations. A positive change represents an increase in area using the E-template based scheme. On average, due to the reduction in wirelengths, 14% and 47% decrease in area was observed with respect to the FDS-VC and Hyper schemes, respectively. In some examples (such as #2, #10) , it was seen that the area increased but the power reduced.

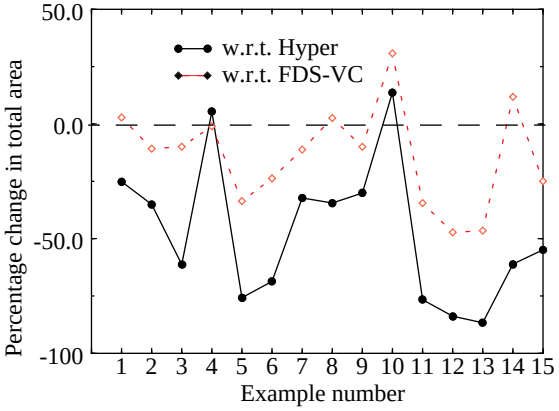


Figure 8. Percentage change in the total chip area.

7. Conclusion

We have presented a new approach to architecture synthesis that targets interconnect (bus, multiplexor, and buffer) power reduction by exploiting the regularity inherent in the algorithm. First, a simple and efficient E-template based assignment and allocation algorithm has been proposed to exploit regularity. Secondly, a modified force-directed scheduling algorithm is used to produce a schedule favorable for regular assignment. Thirdly, a new model is proposed for interconnect length estimation that accounts for the effect of fan-outs on bus lengths.

Our results show that there is a high potential for interconnect power improvements by exploiting regularity inherent in the algorithm. Also, our simple approach is able to capture a large amount of the regularity and results in significant reductions in bus and multiplexor power compared to both the Hyper and the FDS-VC schemes. Reductions are obtained in the total power for all examples and in the overall area for some examples.

8. References

1. R. Mehra, L. M. Guerra, and J. M. Rabaey, “Low Power Architectural Synthesis and the Impact of Exploiting Locality”, *Journal of VLSI Signal Processing*, 1996.
2. M. C. McFarland, “Re-evaluating the Design Space for Register-Transfer Level Hardware Synthesis,” *Proc. of the Int’l Conf. on CAD*, Nov. 1987, pp. 262-265.
3. L. Stok, “Interconnect Optimization for Multiprocessor Architectures,” *Proc. of the IEEE Int’l Conf. on Computer Systems and Software Engg*, May 1990. pp. 461-465.
4. N. Park and F. J. Kurdahi, “Module Assignment and Interconnect Sharing of Pipelined datapaths,” *Proc. of the Int’l Conf. on CAD*, Nov. 1989, pp. 16-19.
5. D.S. Rao and F.J. Kurdahi, "An Approach to Scheduling and Allocation using Regularity Extraction", *Proc. of the European DAC*, 1993, pp. 557-561.
6. M. Corazao, M. Khalaf, L. M. Guerra, M. Potkonjak, and J. M. Rabaey, “Instruction set mapping for performance optimization,” *Proc. of the Int’l Conf. on CAD*, Nov. 1993, pp. 518-521.
7. W. Geurtz, “Synthesis of Accelerator Data Paths for High-Throughput Signal Processing Applications,” *Ph. D. Thesis*, Katholieke Universiteit Leuven, Belgium, Mar. 1995.
8. L. Guerra, M. Potkonjak, and J. Rabaey, “System-level Design Guidance using Algorithm Properties”, *Proc. of the VLSI Signal Processing Workshop*, Oct. 1994, pp. 73-82.
9. P. G. Paulin and J. P. Knight, "Force-Directed Scheduling for Behavioral Synthesis of ASIC's," *IEEE Trans. on CAD*, Vol. 8, No. 6, June 1989, pp. 661-679.
10. M. M. Halldorsson and J. Radhakrishnan, “Greed is Good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs,” *Proc. of the ACM Symp. on the Theory of Computing*, May 1994, pp. 439-448.
11. D. Springer and D. E. Thomas, “New Methods for Coloring and Clique Partitioning in Data Path Allocation,” *Integration, The VLSI Journal*, Dec. 1991, Vol.12, No.3, pp. 267-292.
12. P. E. Landman and J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Trans. on VLSI Systems*, Vol.3, No.2, June 1995, pp. 173-87.
13. A. Masaki, “Possibilities of deep-submicrometer CMOS for very-high-speed computer logic,” *Proc. of the IEEE*, Vol. 81, No. 9, Sept. 1993, pp. 1311-1324.
14. R. Mehra and J. M. Rabaey, "Behavioral Level Power Estimation and Exploration," *Proc. of the Int’l Workshop on Low-Power Design*, April 1994, pp. 197-202.
15. F. J. Kurdahi and C. Ramachandran, "Evaluating Layout Area Tradeoffs for high level synthesis applications", *IEEE Trans. on VLSI systems*, Vol. 1, No. 1, pp. 46-55, Mar. 1993.
16. G. Sorkin, "Asymtotically Trivial Global Routing: A Stochastic Analysis," *IEEE Trans. on CAD*, Vol. CAD-6, No. 5, Sep. 1987, pp. 820-827.
17. J. M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, “Fast Prototyping of Datapath-Intensive Architectures,” *IEEE Design & Test of Computers*, June 1991, pp. 40-51