Improved Low Power FPGA Binding of Datapaths from Data Flow Graphs with NSGA II -based Schedule Selection

Dasanpotty Sai Harish RAM[1], Mugasimangalam Chinnadurai BHUVANESWARI[2], Suresh UMADEVI[1]
[1]*Amrita Vishwa Vidyapeetham University, Coimbatore, 641112, India*
[2]*PSG College of Technology, Coimbatore,641004, India*
ds_harishram@cb.amrita.edu

***Abstract*—FPGAs are increasingly being used to implement data path intensive algorithms for signal processing and image processing applications. In High Level Synthesis of Data Flow Graphs targeted at FPGAs, the effect of interconnect resources such as multiplexers must be considered since they contribute significantly to the area and switching power. We propose a binding framework for behavioral synthesis of Data Flow Graphs (DFGs) onto FPGA targets with power reduction as the main criterion. The technique uses a multi-objective GA, NSGA II for design space exploration to identify schedules that have the potential to yield low-power bindings from a population of non-dominated solutions. A greedy constructive binding technique reported in the literature is adapted for interconnect minimization. The binding is further subjected to a perturbation process by altering the register and multiplexer assignments. Results obtained on standard DFG benchmarks indicate that our technique yields better power aware bindings than the constructive binding approach with little or no area overhead.**

***Index terms*— High level synthesis, Field programmable gate arrays, Power dissipation, Genetic algorithms, Reconfigurable logic**

I. INTRODUCTION

High-level or Behavioral synthesis (HLS) is the process of converting a high-level algorithmic description into an RTL design. The algorithm may be represented in the form of a program or a Data Flow Graph (DFG). The HLS flow consists of the subtasks of *scheduling*, *allocation*, and *binding* which may be executed in any order. However, the result of one subtask shall influence the others. *Scheduling* assigns time-steps to each operation node in the input DFG. *Allocation* fixes the number of functional units or FUs (adders, multipliers etc) and registers for executing the algorithm. *Binding* maps the execution of each operation into a specific FU and also assigns registers for storing the results. Nodes whose execution times do not overlap can share registers. Similarly nodes executing the same operation can share functional units if they are *compatible i.e.*, they do not execute in the same time step. The sharing of functional units necessitates use of multiplexers (MUXes) for selecting the inputs driving the FUs during a particular time step and also selecting the FU output for loading a register. Thus a proper estimate of the area of a binding should take into account the FUs as well as the overhead in terms of registers and MUXes.

In an aggressively low-power design flow, it will be desirable to incorporate power reduction in the HLS stage itself so that different tradeoffs in terms of delay and area can be explored well before the design is implemented in hardware. However, power optimization during behavioral synthesis involves computing the power cost of a candidate solution with the help of simulations and characterizations which might be computationally expensive. In the proposed methodology, an initial GA-based design space exploration phase identifies schedules that are *likely* to yield power efficient bindings. We use an NSGA II based approach [1-4] using an encoding scheme described in [5]. The likelihood of a schedule to produce low-power bindings is used as the power cost of the GA. Actual power estimates are not used.

The schedules identified using the NSGA II step are subjected to a binding step using a *constructive* technique described in [6], which schedules one node at a time. We have adapted this technique for a pre-scheduled graph. Each iteration in the binding process computes the appropriate bit-widths required for the FUs, before allocating FUs, registers and interconnect resources such as MUXes. This is especially important since using a worst case bit-width for a binding leads to over design and thereby wastage of resources and excess area as well as delay for the implementations. The register and FU bindings in each iteration use a weighted bipartite matching approach [6][7] that tries to minimize the incremental area cost of an existing binding when a new node in the DFG is scheduled. The total MUX count is also optimized since steering logic such as MUXes are not implemented efficiently in FPGAs and may contribute significantly to the area and power cost of a binding.

The resultant RTL is subjected to a further *rebinding* step in which the register binding is perturbed with a view to eliminate MUXes or reduce their port count. The RTL obtained after the rebinding step is synthesized to an FPGA target using Xilinx ISE tool and evaluated for power, area and delay. The results on standard DFG benchmarks indicate notable reduction in dynamic power dissipation of the bindings over the constructive method. Schedules obtained from the NSGA run were found to be more amenable to interconnect optimization and the subsequent rebinding step for power reduction than the constructive approach. It is proposed that this approach can serve as the framework for automating the low-power binding process using a transformational heuristic approach. The random

Digital Object Identifier 10.4316/AECE.2013.04015

perturbation can be further improved and used as a mutation operator in a GA or a velocity function in a PSO engine to generate low-power bindings for FPGA targets.

The rest of the paper is organized as follows. Section II reviews literature related to HLS and binding. Section III deals with the encoding of the chromosomes used in the NSGA II run and describes how schedules are generated from a chromosome using a list scheduling heuristic. Section IV describes the two-phase binding methodology. The results of the proposed methodology on standard DFG benchmarks are presented in Section V. Section VI concludes the paper.

II. Review of Related Work

High Level Synthesis (HLS) is an area that has been well researched in the past couple of decades. A survey of HLS techniques is presented in [5]. A GA based scheme for datapath synthesis was proposed in [8]. The technique seeks to optimize FUs, registers and interconnect logic. A bus based interconnection scheme is used in the work. Power is not addressed here. In [9], an Integer Linear Programming (ILP) based approach is used for scheduling and binding of DFGs to FUs. The power cost function is developed by obtaining a curve fit from characterizations for different switching activities. In [10] the authors also propose an ILP approach but combine it with retiming to obtain power optimal bindings. A game theoretic approach is suggested in [11] for obtaining low power bindings.

In [12], the authors describe a methodology for low-power high level synthesis of DFGs targeted at a generic LUT-based architecture. A high-level simulation determines the switching activity between DFG nodes. Interconnect capacitance estimation is carried out using Rent's rule. These parameters are used by a high-level power estimator. The authors propose a simulated annealing approach for iteratively improving an initial schedule towards an optimal solution, guided by the power extracted by the high-level estimator. The solutions obtained after the simulated annealing step are subjected to an additional register binding and port assignment step for optimizing the interconnect resources (MUXes). This is because FPGAs are inefficient in realizing wide MUXes and a binding with fewer numbers of FUs and registers might still incur a large area overhead due to wide MUXes required to route the registers to the FUs. Further, in the port assignment step the connection of registers to FU inputs is revisited to minimize register and MUX usage. The use of fragmentation to reduce useless switching activity in datapaths is proposed in [13]. Binding of multiplications to over-sized FUs leads to unwanted switching activity in the logic. In fragmentation, multiplications of larger bit sizes are implemented by combining smaller multipliers and adders. This makes it possible to implement multiplications with smaller bit sizes using tight fitting FUs, thereby reducing switching activity and dynamic power. A binding scheme with the use of morphable hardware is proposed in [14][15]. The technique exploits the mutual exclusivity between addition and multiplication nodes in DFGs to reconfigure multiplier blocks into adder chains. In [16] a graph based datapath merging approach is employed for scheduling computations in a processor based environment onto runtime reconfigurable hardware that can execute an extended instruction set. In [17], an FPGA synthesis flow is modified to map storage of variables onto RAM blocks in the FPGA with power reduction as the main objective.

In this paper we have adapted a constructive technique for combined binding and scheduling for FPGAs proposed in [6]. The authors incorporate data width also into the synthesis process so that FUs, registers and MUXes of the appropriate sizes are chosen for the binding. Here each node is scheduled and bound based on its *priority*. The node *priority* is computed based on its *mobility*, number of successors and data size. Nodes with zero mobility need to be scheduled in the current time step. Among the rest, nodes with lower mobility are assigned higher priorities. Each scheduled node is bound to an FU based on the minimal incremental path over cost computed from the weighted bipartite graph of the nodes and FUs derived using the method described in [7]. Power reduction is not addressed in this work.

In [18] a multi-objective GA based approach is proposed for area and delay reduction during FPGA behavioral synthesis. A multi-chromosome encoding is used in the work. The assignment field specifies the FU assigned to execute each DFG node. Separate fields are allotted for scheduling, register binding and interconnect binding. These fields contain the algorithm to be used for each. Thus the choice of binding and scheduling algorithms is randomized for better design space exploration.

III. GA Encoding and Scheduling

A. GA encoding

A multi-chromosome encoding [5] is used for the NSGA II phase. The *scheduling priority field* specifies the order in which nodes are taken up for scheduling in a list scheduling heuristic. The *module allocation* field specifies the number of functional units (adders and multipliers) available for executing the DFG nodes. A sample DFG, multi-chromosome and schedule are shown respectively in Figure 1(a), Figure 1(b) and Figure 1(c). The nodes in the DFG are taken up for scheduling in their order of appearance in the chromosome. For example nodes *b0*, *b1* and *a2* are scheduled in the first time step in the schedule in Figure 1(c) since they are the first three nodes in the chromosome.

No more multiplications can be scheduled in this time step since only three multipliers are allocated in the allocation field (2 3). The addition *c0* which appears next in the chromosome can be scheduled only in the next time step since its predecessors *b0* and *a2* are scheduled only in the current step. The algorithm continues in this fashion until all nodes have been scheduled.

B. Power metric

The likelihood of a schedule to yield low-power bindings is determined using a cost function proposed in [1][2]. The cost function is based on the number of edges and edge weights of the *compatibility graph* [19] of a given schedule. The nodes of the compatibility graph or *CG* are the nodes of the DFG themselves. An edge is present between two *compatible* nodes i.e. nodes of the same type that do not execute in the same time step. The weight of the edge is the switching activity when these two nodes execute in

succession in the same FU. The compatibility graph for the schedule in Figure 1(c) is shown in Figure 2

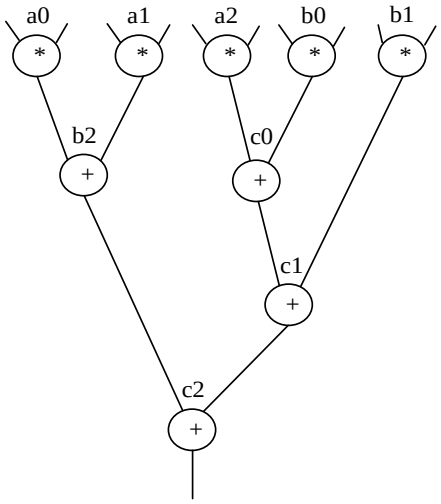

Figure 1(a). Unscheduled DFG

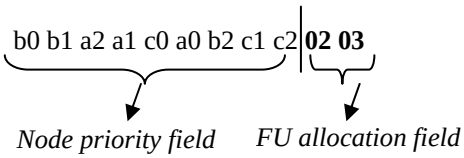

Figure 1(b). Multi-chromosome encoding [5]

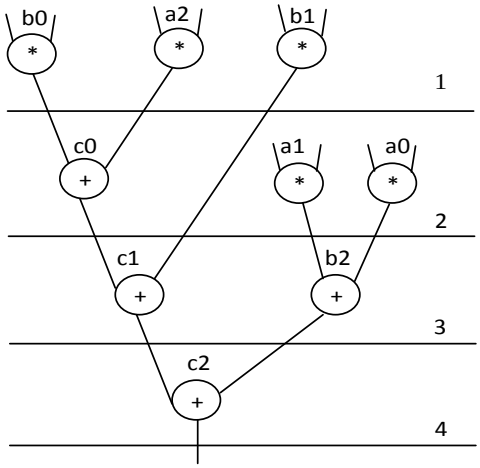

Figure 1(c). Scheduled DFG

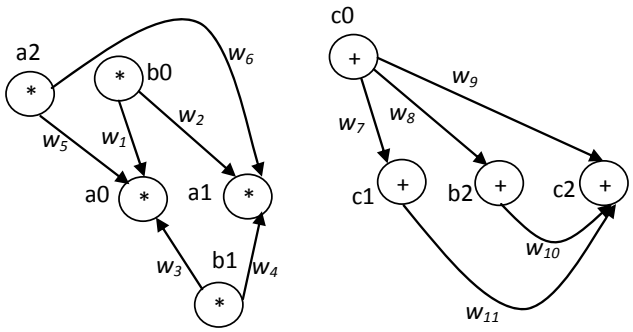

Figure 2. Compatibility graph

Consider the node *a2*. There is an edge from *a2* to *a0* since both are multiplication nodes and are scheduled in different time steps. Thus it is possible to assign the same FU for executing *a0* and *a2*. The weight w_5 indicates the switching cost if *a0* and *a2* execute in succession in the same FU. Note that there is no edge between *a2* and *b0* even though both are multiplications since they are scheduled in the same time step.

The power cost for evaluating a schedule is based on [20] and [1-2]. In [20] the authors establish that schedules that have lesser edge weights in their compatibility graphs are more likely to yield low-power bindings. Also, schedules with a large number of edges in the compatibility graph have a large number of potential bindings possible, thereby increasing the likelihood of bindings whose switching power numbers are close to optimal. Based on these observations in [20], a power metric is proposed in [1-2] which we have adopted for our NSGA II engine and is represented by equation 1

$$P = \frac{n}{m1} + m2 + m3 \tag{1}$$

The term *m1* represents the number of edges. Since it is in the denominator, schedules with higher number of edges (higher value for *m1*) are favoured. The multiplying factor n is user defined. The term *m2* represents the average edge weight of *k %* of the edges where *k* is a tunable parameter whereas *m3* is the average of edge weights of *all* the edges (i.e. *k* = 100 %). Thus it is seen that the cost function favours schedules with higher number of edges in the CG.

IV. Two Phase Binding Methodology

The proposed methodology uses a two-phase approach. The DFG of the algorithm to be synthesized is input to an NSGA II based scheduler [1-4],[21] in the *schedule selection phase*. The schedules with potential to yield low-power bindings are chosen for a subsequent *constructive binding step* followed by a *rebinding* phase involving register reallocation and MUX port reassignment. This is inspired by the transformation operators described in [12][22]. The overall flow is depicted in Figure 3.

A. NSGA-based Schedule Selection Phase

The NSGA II run yields a set of Rank I non dominated solutions that exhibit different trade-offs in power, area and delay. A population of power efficient schedules identified from this set is used as the input to the constructive binding phase

Area cost of the schedule is given by the gate count of the number of FUs and registers determined by the left edge algorithm [21], when synthesized to a generic library. The number of time steps is taken as the delay cost. Power cost of a given schedule is determined in terms of the *likelihood of a schedule to yield low-power bindings* as described in Section III. B. The weights of the edges represent the switching cost of executing the pair of nodes on the same FU. Schedules whose CGs have a large number of edges are likely to yield low-power bindings because of the existence of a number of possible bindings (a given binding maps to a particular combination of paths in the CG). Also, schedules with CGs whose average edge weights are lower, yield bindings that are more power optimal.

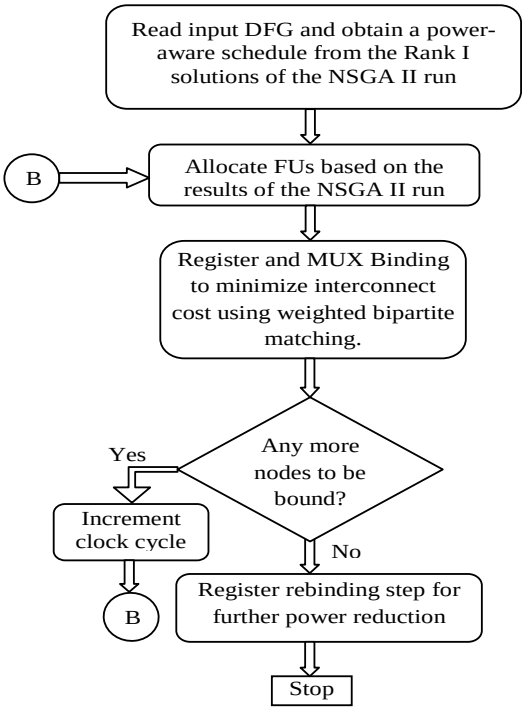

Figure 3. NSGA II based binding methodology

These measures are used to estimate the power cost of a schedule during the NSGA run. The NSGA II scheduler gives a set of *Pareto optimal* solutions that trade-off the parameters of power, area and delay.

B. Constructive Binding and our modified approach

The solutions having CGs with the best power metrics are selected and subjected to a modified version of the constructive binding process [6] for register and interconnect minimization. In this phase each node in the DFG is taken up one at a time for FU allocation based on its *priority*. The priority of a node is calculated based on its mobility and number of successors. Nodes that have zero mobility have the highest priority. The output of a node execution needs to be stored in registers before it is consumed by its successors. Hence nodes with higher number of successors get higher priority since scheduling these nodes earlier frees the output registers for sharing with other nodes. During FU allocation, a node is allocated an available FU based on the *path overcost* i.e., the increase in cost of the MUXes and registers incurred due to the new allocation using a approach based on weighted bipartite graphs [7][23]. The allocation with the minimum path overcost will be selected. In our approach the schedule as well the numbers of FUs are already frozen from the NSGA step. The binding phase is concerned only with the register and interconnects (MUXes). The methodology is summarized in Figure 3.

C. Rebinding for further power reduction

The RTL obtained after binding the power-aware scheduled obtained from the NSGA II run to FUs and registers is further subjected to a *rebinding* process for further power reduction. It was observed that improvement in dynamic power dissipation was achieved in the following cases (i) Ungrouping of multiplication and addition nodes bound to the same register and the resultant MUX elimination (ii) Elimination of MUXes connecting to the registers through register reassignment. The details of the rebinding approaches are given below.

1) Ungrouping of addition and multiplication without adding registers

Addition and multiplication nodes that are compatible may share the same register in the post-binding RTL. The binding is perturbed to ungroup heterogeneous nodes in the same register. This process is further elucidated in Figure 4. A scheduled DFG is depicted in the Figure 4(a) with the node names labeled in bold lower case letters. The FU to which each node is bound is indicated in upper case letters. The registers to which each node is assigned are shown in upper case italics. For instance node *b0* which is a multiplier node is assigned to be executed by the multiplier M1 and the result of executing this node will be stored in the register *R2*. The node *a1* which shares M1 also is bound to register *R2*. Moreover the addition node *c1* executing in adder *A2* also is assigned *R2* for storage. Thus a two input MUX is necessitated for routing the results from M1 and A2 to *R2*

The post binding RTL without application of rebinding is shown in Figure 4(b). The FUs are not explicitly shown for the sake of clarity. The MUX inputs are labeled with the source FUs which drive them. For instance, in the three input MUX in Figure 4(b), the first input is driven by the multiplier M3. The nodes assigned to a particular register are indicated beneath the register itself. For example *R1* is assigned to the nodes *a0*, *a2*, *b2* and *c2*.

Now consider register *R1* which is assigned to the nodes *b2* and *c2* which are addition nodes as well as the multiplications *a2* and *a0*. The register *R4* is assigned to only a single addition node *c0*. Since *b2* and *c2* are executed in the same adder *A1* and their execution times do not overlap with *c0*, they can be reassigned to register *R4* from *R1* without incurring any additional interconnect overhead. This step shall unclutter register *R1* which will be now bound only to the multiplication nodes *a0* and *a2*. This segregation of addition and multiplication nodes bound to registers results in significant switching power reduction since the number of MUX ports is reduced to 2 from 3. The RTL after the rebinding step is shown in Figure 4(c)

It can be seen that the ungrouping has been accomplished without addition of any MUXes that would tend to offset the power reduction achieved. This is because the nodes *b2* and *c2* that are reassigned to the register *R4* from *R1* execute in the same adder *A1* as the node *c0* which was the only node assigned to *R1* initially. The reassignment process leads to a reduction in the number of ports of the MUX connected to register *R4* from three to two since only the multipliers *M3* and *M2* drive the MUX. The reduction in the number of ports contributes to the reduction in dynamic power dissipation.

2) MUX elimination by register rebinding

In this rebinding step, nodes are reassigned to registers with a view to eliminating a multiplexer driving the registers. If a node is reassigned to a register which is already bound to node(s) that share the same FU there is scope for eliminating some of the MUXes. This is illustrated in Figure 5(a) using the same schedule in Figure 4(a) but with a different binding for the FUs.

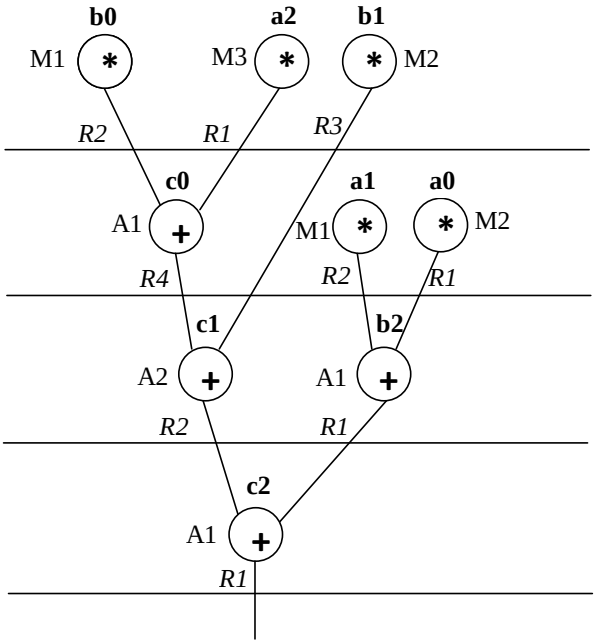

Figure 4(a). A scheduled DFG for illustrating rebinding method 1

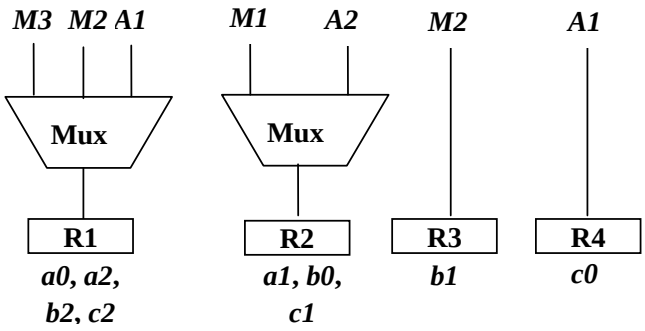

Figure 4(b). Post-binding RTL

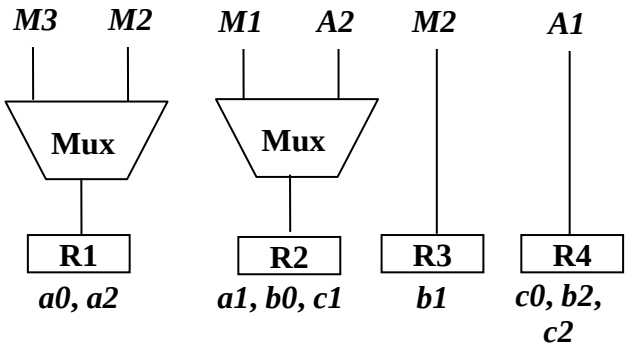

Figure 4(c). RTL after ungrouping

The RTL obtained after binding the above schedule using our modified approach is shown in Figure 5(b). The nodes *a0* and *a2* which are assigned to the multiplier *M2* share the register *R1* with the addition nodes *b2* and *c2*. The nodes *b2* and *c2* can be reassigned to the register *R4* without any additional MUX overhead since they share the same adder as *c0* which is the only node currently assigned to *A1*. This reassignment step also eliminates the MUX for the register *R1* since it is now bound only to the multiplication nodes *a0* and *a2* which are assigned to the same multiplier *M2*. Hence the rebinding step results in the reduction of the number of two input MUXes from two to one. The resultant RTL is shown in Figure 3(c)

The advantages of MUX reduction are two-fold. First, the elimination of MUXes leads to reduction in the switching activity in the datapath thereby significantly reducing the dynamic power dissipation. Also, there is significant reduction in area since MUXes are not efficiently implemented by FPGAs as mentioned earlier.

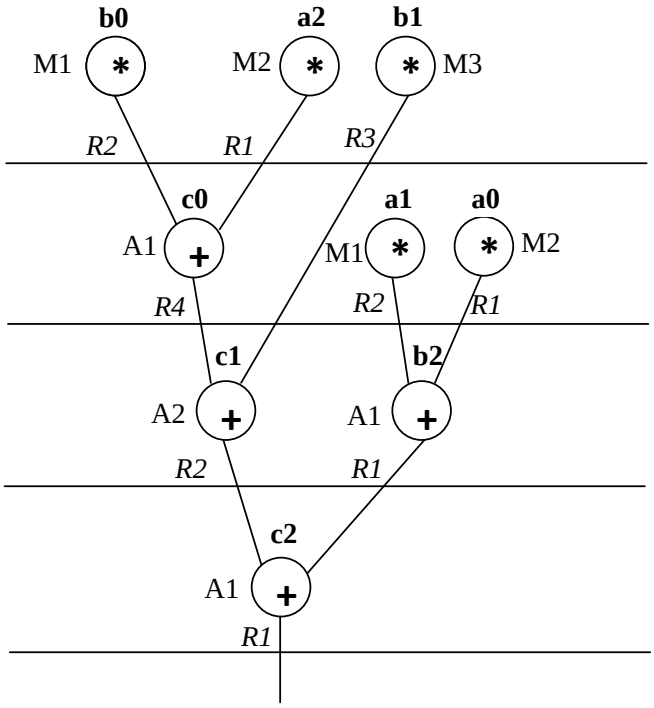

Figure 5(a). Scheduled DFG for illustrating rebinding method 2

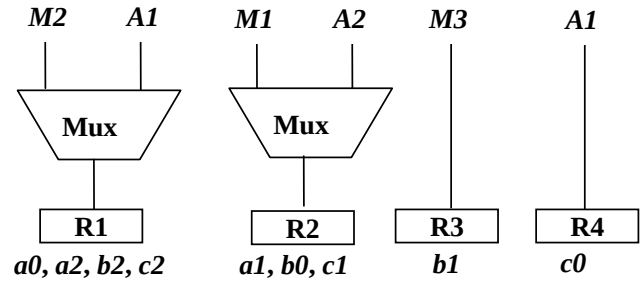

Figure 5(b). Post binding RTL

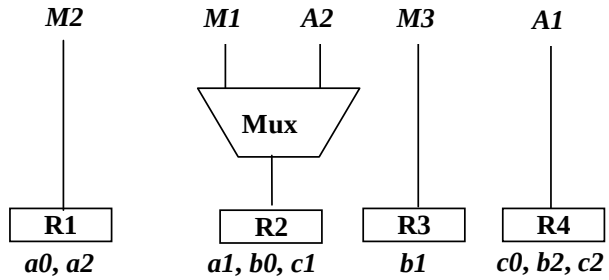

Figure 5(c). RTL after rebinding for MUX reduction

V. Results and Analysis

The binding methodology was evaluated on standard DFG benchmarks. The DFG is input to an NSGA II engine coded in C. Power aware schedules are selected from the Rank I solutions at the end of the NSGA II run. These schedules are capable of yielding bindings with less

switching activity. Also the scheduler ensures that only those schedules that can yield a large number of potential RTL bindings are selected. This provides ample scope for applying the rebinding steps mentioned in Section IV leading to additional savings in switching power. The NSGA II run had a population size of 100 and was run for 100 generations.

Power-aware schedules generated by the NSGA II engine are subjected to our improved binding and rebinding approaches mentioned in Section IV adapted from [6] which seeks to reduce interconnect cost. The binding methodology is implemented in C. The binding tool computes the data widths required for each node in the DFG and allocates the registers and MUXes using the weighted bipartite approach. The area of the MUXes is estimated using the curve-fitted model extracted from Matlab. The final RTL netlist is available as a linked list. The RTL in the form of the linked list is converted into a synthesizable Verilog structural model by a C program.

The structural model of the RTL obtained after the binding and rebinding phases is input to the Xilinx ISE tool and synthesized to a target FPGA library. The dynamic power of the bindings was estimated using the Xilinx Xpower tool. The methodology was evaluated on schedules with different number of time steps. The entire methodology was implemented on a CPU with a i5-2400 Duo processor with 4 GB RAM running at 3.10 GHz. The results are tabulated separately for the FIR, DWT, MPEG and IIR benchmarks from the Mediabench suite [24] in Tables I-IV.

TABLE I. RESULTS FOR THE FIR BENCHMARK

	Constructive approach		NSGA II based approach			Post-rebinding			
Time steps	**Area slices**	**Power (mW)**	**Area (slices)**	**Power (mW)**	**% power reduction**	**Area (slices)**	**Power (mW)**	**% power reduction**	**Rebinding techniques employed**
9	121	163	127	155	5.25%	130	115	29.4%	Method 1 and 2
10	85	150	105	157	Nil	109	92	40.50%	Method 1
15	85	150	62	134	10.60%	74	116	28%	Method 1

TABLE II. RESULTS FOR THE MPEG BENCHMARK

	Constructive approach		NSGA II based approach			Post-rebinding			
Time steps	**Area slices**	**Power (mW)**	**Area (slices)**	**Power (mW)**	**% power reduction**	**Area (slices)**	**Power (mW)**	**% power reduction**	**Rebinding techniques employed**
17	214	530	159	567	Nil	159	469	11.38%	Method 2. No solution with multiplier and adder driving inputs of MUXes
18	214	532	160	584	Nil	166	470	11.58%	- Do -

TABLE III. RESULTS FOR THE DWT BENCHMARK

	Constructive approach		NSGA II based approach			Post-rebinding			
Time steps	**Area slices**	**Power (mW)**	**Power (mW)**	**% power reduction**	**% power reduction**	**Area (slices)**	**Power (mW)**	**% power reduction**	**Rebinding techniques employed**
10	303	183	295	169	7.65%	291	139	24.5%	Method 2. No MUX with adder and multiplier input for applying method 1
11	303	183	259	142	22%	Nil			No MUX associated with registers for applying method 1. No solutions suitable for applying method 2
12	303	182	239	170	6.60%	Nil			No MUX associated with register

TABLE IV. RESULTS FOR THE IIR BENCHMARK

	Constructive approach		NSGA II based approach			Post-rebinding			
Number of time steps	**Area**	**Power (mW)**	**Area (slices)**	**Power (mW)**	**% Reduction in power**	**Area (slices)**	**Power (mW)**	**% Reduction in power**	**Rebinding techniques employed**
4	123	147	127	145	0.93%	117	128	13.11%	Methods 1 and 2
6	92	76	98	71	6.49%	No suitable solutions found for applying rebinding			
7	71	74	73	123	Nil	69	72	3.61%	Methods 1 and 2
8	71	74	75	71	5.14%	No feasible solutions found			

Three different sets of results are presented for each benchmark. The average area and power numbers of the different post-binding RTL solutions computed with Xilinx ISE and Xpower tools [25] are tabulated for schedules with different number of execution steps to exploit the speed-area trade-offs. The first set pertains to the constructive scheduling approach presented in [6]. The second set of results is obtained by applying our binding methodology to power-aware schedules extracted from the Rank I solutions of an NSGA II run. The RTL is further subjected to rebinding using method 1 (ungrouping of multiplication and addition nodes) and method 2 (register rebinding for MUX elimination) described in Section IV. The results of the rebinding processes are shown in the third set of results in each table.

A. Binding of NSGA II schedules

The power cost metric guiding the NSGA II run favours schedules (i) that are likely to produce bindings with low dynamic power and (ii) that have the potential to yield a large number of possible binding solutions thereby increasing chances of RTL datapaths that are near optimal in terms of switching power. There is appreciable power reduction in the case of the DWT, FIR and IIR benchmarks for the bindings of schedules chosen on the basis of the power metric from the NSGA II pool. No reduction in switching power is observed for MPEG. This is due to the algorithm favouring schedules with greater flexibility in binding over switching cost since improvement in the power numbers is observed post-rebinding as explained in Section V. B

B. Rebinding

In the second phase of the binding process, the RTL obtained from the binding of the NSGA II schedules is subjected to a further rebinding step. The rebinding is carried out using the two methods described in Section IV. For the FIR benchmark, appreciable improvement in switching power is noticed for bindings from schedules with different schedule lengths. The last column in the results indicates the techniques that could be employed in the bindings obtained. It can be seen that in some cases solutions suitable for applications of rebinding were not available in the pool of RTL bindings. The bindings for the MPEG benchmark exhibited consistent improvement in dynamic power post-rebinding for all schedule lengths. In the DWT benchmark rebinding was possible only for bindings of schedule length 10 where power reduction was observed. However for this benchmark dynamic power reduction was observed even before rebinding. For the IIR benchmark rebinding was possible only for two schedule lengths wherein marginal power reduction was achieved. This is due to the small number of nodes (9) in the IIR DFG with little flexibility in scheduling and binding and hence less scope for rebinding.

VI. Conclusion and Future Work

A methodology for power-aware binding of datapath schedules from Data Flow Graphs to FPGA targets with the primary objective of reducing switching power has been presented. The technique involves selection of power aware schedules from an NSGA II scheduler which is guided by a power metric that favours schedules with a higher likelihood of yielding low-power bindings and a large number of potential binding solutions. The binding process seeks to minimize interconnect usage during assignment of functional units and registers to various DFG nodes. Results on standard benchmarks indicate appreciable reduction in dynamic power over a constructive approach which schedules and binds one node at a time. The methodology can serve as an efficient rapid design space exploration tool during FPGA synthesis of datapath intensive DFGs where reduction in switching power is an important design objective. Since actual power numbers are not computed during the binding process, computationally expensive low-level characterizations and simulations are avoided. Further work would involve evaluating the technique on additional benchmarks and efforts to further improve the quality of solutions.

References

[1] D. S. H. Ram, M. C. Bhuvaneswari, S. M. Logesh, ”A novel evolutionary technique for multi-objective power, area and delay optimization in high level synthesis of datapaths,” IEEE Computer Society Annual Symposium on VLSI, ISVLSI, pp.290-295, 2011

[2] D. S. Harish Ram, M. C. Bhuvaneswari, Shanthi S. Prabhu, “A novel framework for applying multiobjective GA and PSO based approaches for simultaneous area, delay, and power optimization in high level synthesis of datapaths, VLSI Design journal, vol 2012, doi:10.1155/2012/273276. [Online] Available: http://www.hindawi.com/journals/vlsi/2012/273276/

[3] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Transactions on Evolutionary Computation, vol.6, no.2, pp.182-197, 2002

[4] K. Deb, “Multi-objective optimization using evolutionary algorithms,” John Wiley and Sons, 2003

[5] V. Krishnan, S. Katkoori, A genetic algorithm for the design space exploration of datapaths during high-level Synthesis,” IEEE Transactions on Evolutionary Computation, vol.10, no.3, pp. 213- 229 2006

[6] E. Casseau, B. Le Gal, “High-level synthesis for the design of FPGA-based signal processing systems,” International Symposium on Systems, Architectures, Modeling and Simulation, SAMOS’09, pp.25-32, 2009

[7] C. Y. Huang, Y. S. Chen, Y. L. Lin, Y. C. Hsu, “Data path allocation based on bipartite weighted matching,” Proceedings of the 27th ACM/IEEE Design Automation Conference, DAC ’90, pp 499-504, 1990

[8] Chittaranjan A. Mandal, P. P. Chakrabarti, Sujoy Ghose, “GABIND: A GA approach to allocation and binding for the high-level synthesis of data paths,” Very Large Scale Integration (VLSI) Systems, vol 8 no. 6, pp 747-750, 2000

[9] X. Tang, T. Jiang, A. Jones, and P. Banerjee, “Behavioral synthesis of data-dominated circuits for minimal energy implementation,” Proceedings of the International Conference on VLSI Design, pp 267-273, 2005.

[10] N. Chabini, W. Wolf, “Unification of scheduling, binding and retiming to reduce power consumption under timings and resources constraints,” Very Large Scale Integration (VLSI) Systems, vol.13, no. 10, pp. 1113–1126, 2005.

[11] A. K. Murugavel, N. Ranganathan, “A game theoretic approach for power optimization during behavioral synthesis,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, pp. 1031–1043, 2003.

[12] D. Chen, J. Cong, Y. Fan, L. Wan, “LOPASS: A Low-power architectural synthesis system for FPGAs with interconnect estimation and optimization,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.18, no.4, pp.564-577, 2010

[13] A. A. Del Barrio, S. O. Memik, M. C. Molina, J. M. Mendias, R. Hermida, ”A fragmentation aware high-level synthesis flow for low power heterogeneous datapaths,” Integration, the VLSI journal, doi: 10.1016/j.vlsi.2012.02.005, 2005

[14] D. Bekiaris, E. S. Xanthopoulos, G. Economakos, D. Soudris, “Systematic design and evaluation of a scalable reconfigurable

multiplier scheme for HLS environments," International workshop on communication centric system on chip ReCoSoC, pp 1-8, 2012

[15] D. Bekiaris, G. Economakos, E.S. Xanthopoulos, D. Soudris, "Low-power reconfigurable component utilization in a high-level synthesis flow," International Conference on Reconfigurable computing and FPGAs, pp 428-433, 2011

[16] C. Wolinski, K. Kuchcinski, E. Raffin, F. Charot, "Architecture-driven synthesis of reconfigurable cells," Euromicro conference on digital system design/architecture, methods and tools, pp 531-538, 2009

[17] R. Tessier, "Power-efficient RAM mapping algorithms for FPGA embedded memory blocks," IEEE Transactions on CAD of Integrated Circuits and Systems, vol 26, no.2, 2007

[18] F. Ferrandi, P. L. Lanzi, G. Palermo, C. Pilato, D. Sciuto, A. Tumeo, "An evolutionary approach to area-time optimization of FPGA designs," International Conference on Embedded Systems: Architectures, Modeling and Simulation, IC-SAMOS, pp 145-152, , 2007

[19] J. M. Chang, M. Pedram, "Register allocation and binding for low power," Proceedings of ACM/IEEE Design Automation Conference, DAC '95, pp 29-35, 1995

[20] E. Kursun, R. Mukherjee, S. O. Memik," Early quality assessment for low power behavioral synthesis. Journal of Low Power Electronics, vol 1, no.3, pp 1-13, 2005

[21] S. H. Gerez, "Algorithms for VLSI design automation, John Wiley and Sons, 2000

[22] D. Chen, J. Cong, "Register binding and port assignment for multiplexer optimization. In: ASP-DAC '04 Proceedings of the 2004 Asia and South Pacific Design Automation Conference, pp. 68-73, 2004

[23] R. K. Ahuja, T. L. Magnanti, J.B. Orlin, "Network Flows: Theory, Algorithms, and Applications," Prentice-Hall, Englewood Cliffs, NJ, 1993.

[24] Mediabench benchmark suite Available [Online] express.ece.ucsb.edu/benchmark

[25] Xpower estimator user guide [Online] Available http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/ug440.pdf