Low Power Data Format Converter Design Using Semi-Static Register Allocation

Kala Srivatsan, Chaitali Chakrabarti *

Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-5706
srivatsa@enuxsa.eas.asu.edu, chaitali@asu.edu

Lori Lucke

Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455
lucke@ee.umn.edu

Abstract

In many applications, such as digital signal processing, data format converters are used to reformat the data transferred between processing modules. In VLSI implementations, these converters consume a large portion of the available resources. Various methods have been proposed to synthesize data format converter architectures while optimizing the number of registers used to store the data. In this paper, we present a new register allocation scheme which not only minimizes the number of registers, but also minimizes the power consumption in the data format converter. Low power data format converters are synthesized by minimizing the transitions and interconnections between the registers used to store the data. We present both a heuristic and an integer linear programming formulation to solve the allocation problem. Our method shows significant improvement over previous techniques.

1 Introduction

Data format converters (DFCs) are used to permute the data from one format to another in signal processing and image processing applications. A DFC is used to transpose the data within an algorithm or to reorder the data transferred between heterogeneous modules within an implementation. (The modules within a heterogeneous implementation are assumed to operate on different block lengths and different wordlengths.) Examples of DFCs include matrix transposers, data sequencers, serial to parallel converters, and digit-serial to bit-parallel converters. In this paper we concentrate on the design of low power data format converters. Low power design of DFC architectures is of particular importance since DFCs represent a sizeable portion (20% to 40%) of a VLSI chip, especially for two-dimensional DSP systems. Current industry trends towards low power VLSI circuits mandates a DFC design that minimizes power consumption.

DFCs consist of registers, buses, connection wires, and control circuits. Data transfer between the DFC and processing modules is achieved by the input and output busses. Data transfer between the registers is achieved by inter-register busses. The hardware design of a DFC determines the number of registers to be used, the placement of registers, the interconnections, and the control circuit design. The main sources of power consumption in a DFC are (i) register switching, (ii) register interconnect or the load on a bus, and (iii) control signal generation. The register interconnect consists of the connection between pairs of registers and between the registers and the input-output busses. Recently, techniques have been proposed to synthesize data format converters using the minimum number of registers [1, 2, 3, 8]. The allocation scheme in [1] also attempts to reduce the area overhead due to control circuitry. The method in [2] minimizes the number of registers during high level synthesis. The scheme in [3] minimizes interconnect during allocation of register files for high level synthesis. The sequencer based data path synthesis scheme in [8] tries to take advantage of the regularity of patterns to reduce the number of memory access operations. These methods minimize the number of registers or memory accesses which indirectly minimizes the power consumption. In this paper we directly minimize power consumption by allocating the registers to minimize the factors that contribute directly to the power consumption.

We present a new register allocation scheme which uses the minimum number of registers and directly minimizes the power consumption. Our allocation scheme minimizes the power consumption of DFCs by minimizing the number of register switchings,

*This work was supported in part by a grant from NSF, no. MIP-9309504.

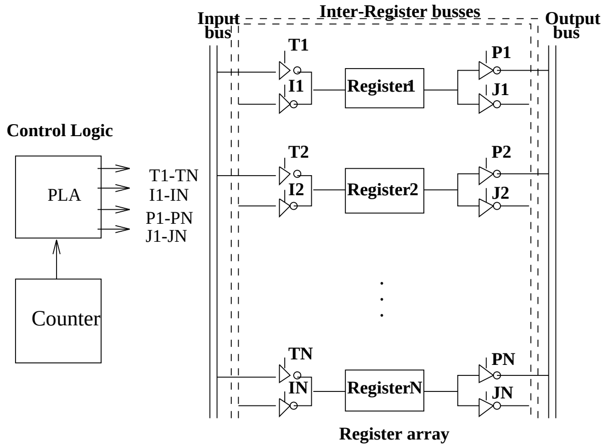

Figure 1: A generic data format converter

minimizing the number of register interconnections, and reducing the complexity of the control circuitry. We minimize the power consumption due to register switching by (a) using minimum number of registers and (b) by minimizing the number of register transitions. We propose a new register allocation scheme called *semi-static allocation*, where each variable is allocated to as few registers as possible. It can be shown that this scheme runs to completion and also sustains interframe pipelining rate as in [1]. We also present an integer linear programming (ILP) model for optimal allocation of variables to registers.

The control circuit for each DFC was implemented by a Programmable Logic Array (PLA) using Mentor Graphics CAD tools. Power consumption in the control circuit is minimized by reducing the number of gates using the Boolean logic optimization procedures available in Mentor Graphics.

The rest of the paper is organized as follows. Section 2 deals with the low power model of a data format converter. The proposed greedy heuristic and the ILP formulation are discussed in Section 3. In Section 4, generation of the control signals for the data format converter is discussed. In addition the power consumption of some data format converters designed by our method are compared with the existing techniques. Concluding remarks are made in Section 5.

2 Power Consumption Model

The proposed data format converter consists of data registers, interconnect, and control as shown in Fig. 1. The registers can read in data from the input bus and the inter-register busses, and can place data on the output bus and the inter-register busses. Each register in Fig. 1 is actually several bits wide. The width of a register depends on the data wordlength. In this paper we concentrate primarily on minimizing the power consumed in the registers and their interconnections.

The power consumption of a CMOS VLSI circuit has been modelled in [5, 6]. The power, P, is defined as $P = 0.5 f_c C_l V_{dd}^2$, where f_c is the clock frequency of the circuit, C_l is the effective load capacitance, and V_{dd} is the power supply voltage. In the data format converter, the power consumption of the registers can be estimated as $P = 0.5 f_c f_d R C_{latch} V_{dd}^2$, where f_d is the activity factor of the latch, R is the number of registers, and C_{latch} is the capacitance of a latch [6].

In the data converter, the power consumption in the registers is proportional to $C_{latch} f_d$. In the proposed scheme, the registers are built with static latches, resulting in a C_{latch} value that is twice that of the dynamic registers in [1]. The savings scheme comes from a significantly smaller value of f_d. In [1], $f_d = 1$ since the variables move continuously from one register to another. In our scheme we minimize the number of register transitions resulting in a small value of f_d. This feature results in significant reduction in the power consumption due to register switching.

3 Low Power Register Allocation Scheme

In this section, we propose two methods to design low power data format converters. Both methods achieve low power design by minimizing two factors: number of register transitions and number of split variables. Minimizing the transitions reduces the activity factor while minimizing the split variables reduces the register interconnect and the control complexity.

The first method presented in this section relies on a heuristic procedure that tries to simultaneously minimize all the above factors to obtain a near optimal solution. The second method is based on an integer linear programming formulation and yields the optimal solution to the low power design problem.

3.1 Proposed Heuristic

The proposed heuristic minimizes the number of transitions of any particular variable by allocating that variable to as few registers as possible. In addition, the number of variables undergoing transitions is minimized by grouping variables based on their lifetimes and allocating the groups to registers. The heuristic algorithm is described below. Let P be the period. The period is the number of time steps necessary to input all the variables for one data conversion.

Variable	*Birth*	*Death*	*Period*	*Step4*	*Step5*
a	0	9	9		R4
b	1	13	12		R5
c	2	17	15		R2
d	3	21	18	d_1R1	d_2 R6
e	4	10	6		R7
f	5	14	9		R8
g	6	18	12		R6
h	7	22	15		R3
i	8	11	3		R9
j	9	15	6		R4
k	10	19	9		R7
l	11	23	12		R9
m	12	12	0		-
n	13	16	3		R5
o	14	20	6		R8
p	15	24	9		

Table 1: (4×4) sequential matrix transposer: execution trace of the proposed algorithm

Let L_i and D_i be the birth time and death time of variable i.

Algorithm:

- Step 1: Find the minimum number of registers using lifetime analysis.
- Step 2: Divide the variables in to three groups such that group (I) consists of variables with lifetimes equal to P, group (II) consists of variables with lifetimes less than P and group (III) consists of variables with lifetimes greater than P.
- Step 3: Assign variables in group (I) directly to individual registers. Since all the variables in this group have time period equal to P, each variable is assigned to a different register. The number of registers used for allocation equals the number of variables in this group. Update the available timeslot after this assignment.
- Step 4: Split each variable in group (III) into two variables: one with lifetime equal to or less than period, P, and the other with the remaining lifetime. Repeat Step 3 on the variables with lifetimes equal to P. Update the available timeslot. The unassigned variables in Step 4 are assigned in the next step.
- Step 5: Sort the group (II) variables and unassigned variables from Step 4 in decreasing order of their lifetimes. Using this sorted list, collect variables into subgroups such that no two variables in a subgroup have overlapping lifetimes and the sum of the lifetimes of all the variables within a subgroup is less than P. An ideal case for a subgroup would be if the death time of each variable is the birth time of some other variable and the combined lifetime of all variables equals P. Sort the subgroups in decreasing order of the sum of the lifetime values of all variables contained in those subgroups. Allocate the subgroups from the sorted list to registers. Update the available time slot. Variables that could not be sorted into subgroups are allocated in Step 6.
- Step 6: Assign the variables in the available time slots in decreasing order of their lifetimes. Update the available timeslot after each assignment. Repeat this step till all the variables are allocated to registers.
- Step 7: Regroup the variables in a different way and repeat Steps 4, 5 and 6 if more than one variable gets split. Repeat steps 4, 5 and 6 till the minimum number of variables are split.

Table 1 lists the register allocation of the 4×4 sequential matrix transposer from [1] based on the proposed heuristic. In this example, the minimum number of registers determined from lifetime analysis is 9 and the period is 16 time units. There are no variables in group (I) and hence Step 3 is not applicable. In Step 4, the variable d can be split into two variables d_1 and d_2 by more than one method. One way of splitting d results in d_1 with birthtime of L_i and other way results in d_1 having deathtime D_i. d_1 is allocated to register $R1$ in this step and d_2 is carried to the next step. In Step 5, the variables c and h are allocated to registers R2 and R3 as their lifetime is nearly equal to P. The other group (II) and unassigned group (III) variables are grouped based on their lifetimes. Each method of splitting may lead to different results so repeat steps 3 to 6 to get an optimal result.

In the first iteration, variable d was split into d_1 with lifetime $(5 - 21)$ and d_2 with lifetime $(3 - 5)$. This allocation resulted in 8 transitions and 2 variable splits. The variables d_2 $(3-5)$ and g $(6-18)$ are allocated to register 6 in a non-ideal way (the birth time of the second variable was not equal to the death time of first variable). In the second iteration, variable d was split in a different way with variables d_1 and d_2 having lifetimes $6 - 21$ and $3 - 6$ respectively. This allocation not only resulted in an ideal case for allocation to register 6 but also only one variable p was split for the same number of transitions. In Step 7, the variable p with lifetime from $15 - 24$ was assigned among the available time slots.

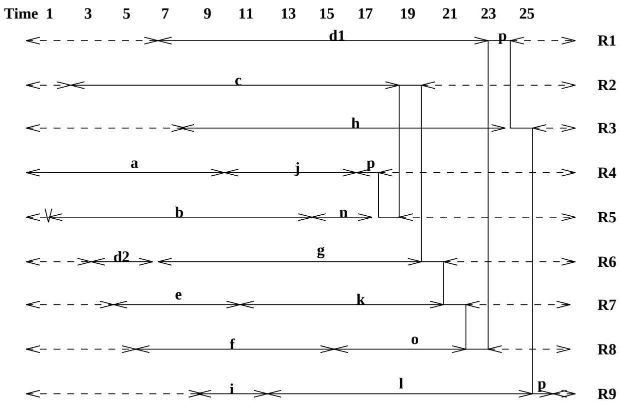

Figure 2: (4×4) sequential matrix transposer: assignment of variables by the proposed algorithm

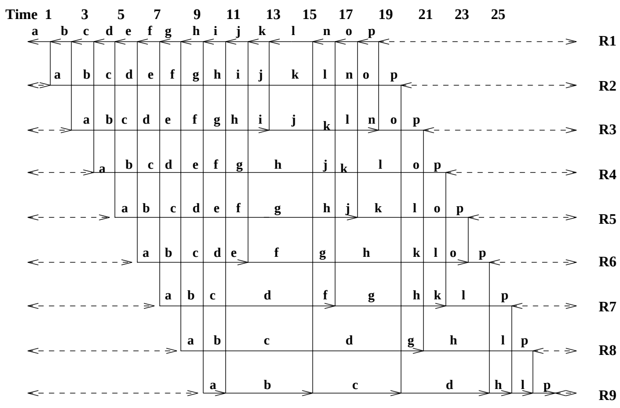

Figure 3: (4×4) sequential matrix transposer: assignment of variables by Parhi's algorithm

Fig. 2 shows the assignment of variables to registers by the proposed method. For comparison, the allocation of variables to registers using the technique in [1] is shown in Fig. 3. It is obvious from the figures that the proposed technique significantly reduces the number of variable transitions between registers.

Table 2 lists the number of transitions obtained for the 4×4 sequential matrix transposer example by the different methods. Table 3 lists the number of variable splits. The number of transitions and total number of variables getting split in the proposed allocation scheme are less compared to other methods. Minimizing the number of variables getting split reduces the complexity of the control circuit and reduces the number of inter-register busses. In the above example only one variable undergoes transitions at any instant of time and hence the load on the inter-register bus is reduced. In fact, the inter-register bus in this case can

DFC	*[1]*	*[8]*	*heur.*	*ILP*
4×4 seq-transposer	174	64	39	37
1D-DWT	57	18	13	13
$(2,1) \rightarrow (3,1)$converter	133	72	34	34
$(3,1) \rightarrow (1,2)$converter	132	60	23	23
4×4 par-transposer	192	72	33	33
Sequencer for FIR filter	80	56	32	32

Table 2: Comparison of the number of register transitions

DFC	*[1]*	*[8]*	*heur.*
4×4 seq-transposer	15	15	2
1D-DWT	6	6	1
$(2,1) \rightarrow (3,1)$converter	18	18	0
$(3,1) \rightarrow (1,2)$converter	12	12	1
4×4 par-transposer	15	15	3
Sequencer for FIR filter	14	14	4

Table 3: Comparison of the number of splitting variables

be eliminated by connecting the registers in a chain. The resulting power reduction is significant.

3.2 ILP Formulation

In this section, we describe an ILP model for optimally allocating variables to registers. This ILP model finds a register allocation that minimizes total power consumption by modelling transition minimization, and variable split minimization. We define the following parameters for the ILP model.

I and J denote the set of variables and registers respectively. K denotes the total number of time steps and P denotes the period. Note that K is larger than P since the schedule for the allocation of registers overlaps from one period to the next.

A variable $i \in I$ exists between L_i and D_i where L_i is the birth time and D_i is the death time.

$x_{i,j,k}$ denotes a binary variable that has a value of 1 if variable i is assigned to register j at time k and has a value of 0 otherwise.

$y^1_{i,k}$ denotes a binary variable that takes on a value 1 if variable i switches to a higher number register from time k to time $k+1$ and takes on a value of 0 otherwise.

$y^2_{i,k}$ denotes a binary variable that takes on a value 1 if variable i switches to a lower number register from time k to time $k+1$ and takes on a value of 0 otherwise.

S_i denotes a binary variable that takes on a value of 1 if variable i splits during its lifetime and takes on a value of 0 otherwise.

The ILP model minimizes the power consumption while satisfying the following constraints.

$$\text{Minimize COST} = C_1 \sum_i S_i + C_2 \sum_i \sum_k y^1_{i,k} + C_2 \sum_i \sum_k y^2_{i,k}$$

$$\sum_j x_{i,j,k} = 1 \text{ for } i \in I, j \in J, L_i \le k \le D_i \quad (1)$$

$$\sum_i x_{i,j,k} \le 1 \text{ for } i \in I, j \in J, 0 \le k \le K \quad (2)$$

$$My^1_{i,k} + \left(j \sum_j x_{i,j,k} - j \sum_j x_{i,j,k+1} \right) \ge 0 \quad \text{for } i \in I, j \in J, L_i \le k \le D_i - 1 \quad (3)$$

$$My^2_{i,k} + \left(j \sum_j x_{i,j,k+1} - j \sum_j x_{i,j,k} \right) \ge 0 \quad \text{for } i \in I, j \in J, L_i \le k \le D_i - 1 \quad (4)$$

$$\sum_j x_{i,j,k} + \sum_j x_{i,j,k+P} \le 1 \quad \text{for } i \in I, j \in J, k + P \le D_i \quad (5)$$

$$MS_i - \left(\sum_k y^1_{i,k} + \sum_k y^2_{i,k} \right) \ge 0 \quad \text{for } i \in I, j \in J, L_i \le k \le D_i - 1 \quad (6)$$

The cost function in the ILP formulation is a function of the transition minimization and variable split minimization. Transition minimization is achieved by including the terms $y^1_{i,k}$ and $y^2_{i,k}$ in the COST function of the ILP. The number of variables that are split is minimized by the term S_i in the COST function. The results presented at the end of this section are based on a simplistic assignment of $C_1 = C_2 = 1$. Needless to say that a better allocation would be obtained by calculating realistic values of C_1, C_2 from circuit simulations.

DFC	*[1]*	*[8]*	*heur.*	*ILP*
4×4 seq-transposer	1	0.36	0.22	0.21
1D-DWT	1	0.31	0.22	0.22
$(2,1) \rightarrow (3,1)$converter	1	0.54	0.25	0.25
$(3,1) \rightarrow (1,2)$converter	1	0.45	0.17	0.17
4×4 par-transposer	1	0.38	0.17	0.17
Sequencer for FIR filter	1	0.70	0.40	0.40

Table 4: Comparison of activity factors

The significance of equations (1) through (6) are as follows. Constraint 1 ensures that each variable is assigned exactly to one register during each time step. Constraint 2 ensures that a register can have a value of 1 or 0 during each time period. Constraint 3 checks for transition of a variable from a lower number register to a higher number register at each time step. Here M is a predefined large number. Similarly Constraint 4 checks for transition of a variable from a higher number register to a lower number register at each time step. Constraint 5 is the period constraint which ensures that if a variable is allocated to a particular register at time k then no other variable is allocated to the same register at a time $k+P$, where P is the period. This constraint is very important for repetitive DFCs. Constraint 6 reduces the total number of variables getting split. The ILP models were solved using the GAMS/OSL solver [9].

Table 2 compares the number of register transitions obtained for each of the existing methods [1], [8] with the proposed heuristic and ILP formulation. Table 3 compares the number of variable splits and Table 4 compares the activity factors of the proposed methods with those of [1], [8]. In [1] every register is making a transition at every clock. Thus the activity factor is one. The activity factors for the other techniques are calculated as the ratio of measured transitions divided by the number of transitions used in [1]. Table 5 calculates the power reduction of the proposed methods compared to the method in [1]. The power reduction is calculated based on the activity factors from Table 4. These power calculations are based on the number of register transitions and do not include bus loading.

4 Control logic

In this section we describe the control logic for the proposed register allocation scheme. For the 3×3 sequential transposer, 13 different states were required to generate the control signals in [1] and [8], as well as the proposed greedy heuristic. A PLA model was synthesized using Mentor Graphics software for gener-

DFC	*[1]*	*heur.*	*ILP*
4×4 seq-transposer	0	56%	58%
1D-DWT	0	56%	56%
$(2,1) \rightarrow (3,1)$converter	0	50%	50%
$(3,1) \rightarrow (1,2)$converter	0	66%	66%
4×4 par-transposer	0	66%	66%
Sequencer for FIR filter	0	20%	20%

Table 5: Power reduction based on calculated activity factors

DFC	*[1]*	*[8]*	*heur.*
3×3 seq-transposer:			
5 input AND	23	23	19
5 input OR	8	9	7
Inverter	23	25	19
Total	54	57	45

Table 6: Total number of gates in the control circuit

ating the control signals. In all cases, logic minimization was done using Autologic. Table 6 compares the number of gates used in the control circuit of [1] and [8] with the proposed heuristic. Note that the number of gates required by the proposed method is less compared to [1] and [8].

5 Conclusion

A greedy heuristic and ILP formulation for low power register allocation of DFCs has been proposed. The experimental results for a number of DFCs show encouraging results. Our design methodology results in both low power consumption and low area designs. The low power consumption is achieved by minimizing the transitions and control circuit components. Area efficient designs are achieved by using minimum number of registers and reducing the number of gates in the control circuitry.

In the current model we calculate the power consumption based on the number of register transitions. This is used to calculate the activity factor. A more accurate model of the activity factor would account for the power consumption due to the loading on each bus. In future models we plan to directly account for the loading on each bus. We plan to combine the transition minimization, interconnect minimization, and control minimization models to provide a global power minimization model.

5.1 Acknowledgments

We would like to thank Dr. N. Srivatsan for his with the ILP formulation.

References

[1] K. K. Parhi, "Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation," *IEEE Trans. on Circuits and Sys.-II*, Vol. 39(7), pp. 423-440, 1992.

[2] K. Ito, and K. K. Parhi, "Register minimization in cost-optimal synthesis of DSP architecture," *Proc. of the IEEE VLSI Signal Processing Workshop*, Sakai, Japan, Oct. 1995.

[3] L. Stok and J. Jess, "Foreground memory management in data path synthesis," *Int'l Journal of Circuit Theory and Applications*, Vol. 20, pp. 235-255, 1992.

[4] K. Ito, L. Lucke, and K. Parhi, "Module selection and data format conversion for cost-optimal DSP synthesis," *Proc. of IEEE/ACM Conf. on Computer-Aided Design*, 1994.

[5] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low power CMOS digital design'," *IEEE Jour. of Solid State Circuits*, Vol. 27(4), 1992.

[6] D. Liu and C. Svensson, "Power consumption estimate in CMOS VLSI chips," *IEEE Jour. of Solid State Circuits*, Vol. 29(6), pp. 663-670, 1994.

[7] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. on VLSI Systems*, Vol. 1(2), pp. 191-202, 1993.

[8] M. Aloqeely and C. Y. Roger Chen, "Sequencer based data path synthesis of regular iterative algorithms," *31st DAC proceedings 1994(IEEE Cat. No. 94CH3408-2)*, pp. 155-160, 1994.

[9] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, San Francisco, CA: The Scientific Press, 1992.