Data Reuse Exploration under Area Constraints for Low Power Reconfigurable Systems

Qiang Liu*, George A. Constantinides*, Konstantinos Masselos† and Peter Y.K. Cheung*
*Department of Electrical and Electronics Engineering
*Imperial College, London SW7 2BT, U.K.
†University of Peloponnese, Tripolis, Greece
{qiang.liu2, g.constantinides, k.masselos, p.cheung}@imperial.ac.uk

ABSTRACT

Contemporary FPGA-based reconfigurable systems have been widely used to implement computation-intensive applications. In these applications data transfer and storage consume a large part of the system energy. Exploiting data reuse can introduce significant power savings, but also introduces extra requirement for on-chip memory. To aid data reuse design exploration, we present a optimization approach to achieve a power-optimal design satisfying area constraints in a targeted reconfigurable platform. The data reuse exploration problem is mathematically formulated and shown to be equivalent to a Multiple-Choice Knapsack problem. The solution to this problem corresponds to the decision of which arrays to be introduced where in order to minimize power consumption for a fixed on-chip memory size. We also present an experimentally verified power model, capable of providing the relative power information between different data reuse options, resulting in a fast and efficient design space exploration. The experimental results demonstrate that the approach enables us to find the most power efficient design for each benchmark.

1. INTRODUCTION

Contemporary FPGA-based reconfigurable systems have been widely used to implement computation-intensive applications. This is ascribable to high flexibility, embedded heterogeneous hardware resources and easy upgrade of the systems. Computation-intensive applications, such as real-time digital signal processing, usually involve a large number of data transfers between the processor and off-chip memories that could consume 50-80% of the embedded system energy [15]. Therefore, design space exploration aimed at reducing the power portion consumed in memory accesses is highly necessary to improve the system energy dissipation.

To reduce accesses to large off-chip memories, which are hungry for power, repeatedly used data can be stored in on-chip memories with low power consumption per access, known as *data reuse* [5]. An approach for exploiting data reuse in FPGA-based platforms has been proposed in [11], where arrays buffering reused data are introduced for each array reference in a nested loop and the results show that keeping frequently used data in FPGA on-chip memories results in a significant reduction of the off-chip memory accesses. The work in this paper is to refine the data reuse decision step of the approach [11], based on balancing the power consumption and on-chip memory resources. In this paper we consider FPGA on-chip dedicated RAMs as on-chip memories and reused data are only stored in them, *i.e.* there are only two memory levels, off-chip RAMs and on-chip dedicated RAMs, in the target platform. While we target the FPGA-based platform, the proposed optimization approach can be readily extended to application specific environments with multiple memory levels.

Clearly, the more reused data are buffered locally, the more reduction of off-chip memory accesses can be achieved. At the same time, the required on-chip memory resources are increased, leading possibly to increase of the size of the required FPGA device, as well as the FPGA on-chip power consumption. These can be seen in a simple example in Fig. 1 (a). Because of two memory levels, in this example there are at most four data reuse options, in each of which an array mapped to on-chip dedicated RAMs could be introduced to buffer reused data at one of the points RA_0–RA_3 for the off-chip large array *image*. According to the approach in [11], an option without data reuse and two beneficial data reuse options [1] are presented in Fig. 1 (b), (c) and (d). Assuming the image is QCIF frame size, the number of off-chip memory accesses of the implementation in (b) is 228096. when data reuse is exploited in (c) and (d), the number of off-chip memory accesses are 76032 and 25344, respectively, with requirements of 1 and 16 blocks of on-chip dedicated RAMs. We can see that a reduction of the number of accesses to the external memories of 9 times can be obtained at the cost of 16 embedded dedicated blocks of RAMs. Obviously, if the targeted FPGA is unable to provide the required on-chip memory resources or the increased power caused by the additional on-chip resources exceeds the power reduced by the optimization of the external memory accesses, then the corresponding data reuse option is useless. Therefore, exploration of the data reuse design space to minimize the system power consumption, given the FPGA on-chip area constraint, is needed and is a laborious work. Especially, when there are multiple array references and each of them owns multiple data reuse options, the number of potential options grows exponentially to the number of array references. This paper addresses a practical approach to solve this problem automatically.

Specifically, the main contributions of this paper are:

- a formalization of the problem of exploring data reuse space with a low power objective under an on-chip area constraint,

[1] A beneficial data reuse option is an option in which for each array buffering reused data the number of reads from this array is larger than the number of writes to it.

```
/* RA0 */
For (x = 0; x < N; x++)
  /* RA1 */
  For (y = 0; y < M; y++)
    /* RA2 */
    For (i = -1; i < 2; i++)
      /* RA3 */
      For (j = -1; j < 2; j++)
        ... = image [ (x + i) * M + y + j ] ...;
```
(a)

```
For (x = 0; x < N; x++)
  For (y = 0; y < M; y++)
    For (i = -1; i < 2; i++)
      For (j = -1; j < 2; j++)
        ... = image [ (x + i) * M + y + j ] ...;
```
(b)

```
For (x = 0; x < N; x++)
  Data_loading (RA1 , image);
  For (y = 0; y < M; y++)
    For (i = -1; i < 2; i++)
      For (j = -1; j < 2; j++)
        ... = RA1 [ i * M + y + j ] ...;
```
(c)

```
Data_loading (RA0 , image);
For (x = 0; x < N; x++)
  For (y = 0; y < M; y++)
    For (i = -1; i < 2; i++)
      For (j = -1; j < 2; j++)
        ... = RA0 [ (x + i) * M + y + j ] ...;
```
(d)

Figure 1: The code segment of Sobel edge detection algorithm. (a) The code commented with possible data reuse insertions at RA_i, (b) Implementation without data reuse, (c) Implementation with a data reuse array introduced at RA_1, (d) Implementation with a data reuse array introduced at RA_0.

- a simple and experimental power consumption model for FPGA-based reconfigurable systems,
- the recognition that, under this power model, the problem is equivalent to the well-known Multiple-Choice Knapsack problem [12],
- an application of the proposed approach to several benchmarks, resulting in the optimal designs for each of them in terms of the power consumption and area.

The rest of the paper is organized as follows. In Section 2 we briefly describe related research work. Section 3 formulates the problem of data reuse exploration. In Section 4 we propose a dynamic power model. In Section 5 we present experimental results of the application of this approach to real benchmarks and conclude in Section 6.

2. BACKGROUND

Research work aimed at achieving the optimal memory configuration has been extensively carried out in embedded system designs. Extending previous work, Shiue *et al.* [15] explore data cache configurations based on cache size, performance and energy consumption. They use the average memory access time and the memory energy consumption as the performance metric and energy metric, respectively. This simplifies the estimations of the performance and energy consumption, and makes this approach feasible in the early design stages. Petrov *et al.* [13] propose a customization approach to partition the data cache and only buffer the data of the array references exhibiting data reuse in the same partition, in order to reduce the cache miss rate and improve the performance and the power consumption. A well designed hardware controller is needed to support this scheme and comes as an overhead. A systematic methodology for data reuse exploration in embedded systems is presented in [5]. A cost function of power and area of the memory system is used to choose the best promising memory hierarchy. The effects of different combinations of data reuse transformations and memory system structures on the system power consumption and performance have been shown in [6] and [16] for the platforms with multiple embedded instruction set processors.

Techniques have been proposed in [1] and [9] for scratchpad memory (SPM) to analyse the data reuse existing in the nested loops and determine which level of the nested loop the data reuse is exploited and which elements of the arrays are stored in the SPM. To make this decision, a memory access cost model, which considers the costs taken by accessing to the external memories and to the SPM, is used. The cost could be power or execution time.

There exist some research works on buffering reused data in FPGA on-chip dedicated RAMs and registers [3, 17]. They use the following simple algorithm to do the data reuse exploration. For an array where elements are accessed several times, it is firstly assigned to be stored in registers if there are enough available registers. Otherwise, it is stored in embedded dedicated RAMs. In case there are not enough on-chip dedicated RAMs left, this array can be only accessed from the external memories. Baradaran *et al.* [2] select arrays more beneficial to minimize the memory access time and store them in the limited amount of registers.

Our approach, focusing on data reuse exploration for SPM in reconfigurable systems, differs from previous work described above mainly in two aspects. Firstly, the power consumption model used in the approach is simpler and parameterizable; we have experimentally determined the parameters for certain Xilinx FPGAs. This model roughly estimates the power, but, crucially, it is accurate enough to know the *relative* merit among different data reuse options at compile time in order to choose the promising options. Secondly, we mathematically formulate the data reuse exploration problem and evaluate overall options for all array references to obtain the optimal solution. This exploration problem is NP-hard but we show in this paper that it is equivalent to the Multiple-Choice Knapsack problem. As a result, existing exact and approximation algorithms can be used to find optimal or feasible solutions to this problem.

3. FORMULATION OF THE DATA REUSE EXPLORATION PROBLEM

The proposed data reuse approach targets n-level loop nests surrounding multiple array references. For each reference, there are n possible places where a single on-chip array can be inserted to buffer reused elements of the reference. For the example in Fig. 1 (a), there are four possible reuse points for the array *image* in the four-level nested loop. Thus, there are in total $n + 1$ design options for a reference, including n possible data reuse options and an option without introduction of an on-chip buffer. Then, for m references, the candidate pool has $(n+1)^m$ options, which results in a very large search space for modest m. Choosing the best option from the large pool with the objective to minimize the system power consumption under the on-chip area constraint can be formulated as an Integer Linear Programming (ILP) problem.

For the presentation in this section, we assume the following notations. There are m array references (loads/stores) A_i $(1 \leq i \leq m)$ and reference A_i owns k_i options for on-chip buffer placement OP_{ij} $(1 \leq j \leq k_i)$. Here, k_i may be less than $n+1$ because some data reuse options could be illusive, *e.g.* the number of reads from the buffer is not larger than the number of writes to it in the possible options.

Each OP_{ij} will consume power P_{ij}, including on and off-chip power and the overheads required to load the on-chip buffer, as calculated in Section 4, and occupy B_{ij} blocks of on-chip dedicated RAM, a value that can be calculated following the methodology in [10]. Then, the problem can be defined as the following:

$$\min : \sum_{i=1}^{m} \sum_{j=1}^{k_i} P_{ij} x_{ij} \quad (1)$$

subject to

$$\sum_{i=1}^{m} \sum_{j=1}^{k_i} B_{ij} x_{ij} \leq B \quad (2)$$

$$\sum_{j=1}^{k_i} x_{ij} = 1, 1 \leq i \leq m \quad (3)$$

$$x_{ij} \in \{0,1\}, 1 \leq j \leq k_i, 1 \leq i \leq m \quad (4)$$

The objective function, minimization of the system power consumption, is given in (1). The inequality (2) defines the area constraint, where there are B blocks of on-chip dedicated RAMs available. Variables x_{ij} can only be set to zero or one in (4). Here, x_{ij} taking value one means the option OP_{ij} is selected for the reference A_i, zero otherwise. Finally, equation (3) indicates that exactly one option is chosen for each reference.

This ILP problem could be solved with general purpose ILP software, however there is a correspondence between this problem and the so-called Multiple-Choice Knapsack (MCK) problem [12]. MCK is the following problem. Given multiple classes of items to pack in a knapsack with a capacity limit and each item having a profit and a weight, one item is chosen from each class such that the profit sum is maximized and the weight sum does not exceed the capacity limit. We can take each array reference in the data reuse exploration problem to be a class and options of this reference as items belong to this class, the negative of the power consumption ($-P_{ij}$) and the number of blocks of on-chip RAM (B_{ij}) as the profit and the weight of each item respectively. Then, it can be noted that our problem is, in fact, equivalent to the Multiple-Choice Knapsack problem.

There exist several algorithms to solve the Multiple-Choice Knapsack problem. Therefore, if P_{ij} and B_{ij} of each option OP_{ij} can be evaluated at compile time, then a solution to the data reuse exploration problem can be obtained. The number of blocks of dedicated RAMs, B_{ij}, taken by an option is decided by the size of the array buffering reused elements of the reference, which can be estimated by analyzing the surrounding loop structure [10]. However, an accurate power estimation for each option in the high level synthesis stage is difficult. Therefore, we focus on the development of such a power model.

4. PROPOSED POWER MODEL

There exist several power models for the embedded systems. Landman's energy model has been used in [5, 6, 16], which is parameterized by the capacitances and frequencies of read and write operations. Several power models exist for caches [15], which are based on estimation of the cache line hit rate. Also some memory access models have been addressed in [1, 9], which take costs of transferring data between off-chip memory and SPM and of accessing to the SPM into account. The model used in our approach is to estimate the dynamic system power of the FPGA-based reconfigurable systems with external SRAMs.

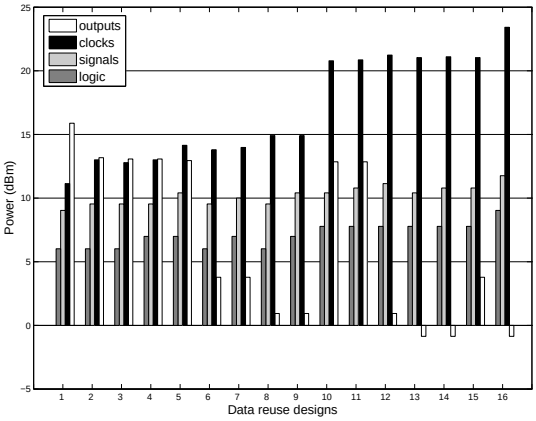

Figure 2: The composition of the on-chip dynamic power of the FSME implementations.

Generally, total system power consumption consists of dynamic and static parts. Given a hardware platform, the static power consumption keeps constant, thus we focus on the dynamic part. The dynamic power consumption P_{d} for our target FPGA-based computing platform mainly consists of two parts: the power consumed in off-chip memory P_{off} and in on-chip P_{on}. If P_{off} and P_{on} can be estimated separately at compile time, then the dynamic power consumption P_{d} can be estimated by:

$$\widehat{P}_{\mathrm{d}} = \widehat{P}_{\mathrm{off}} + \widehat{P}_{\mathrm{on}}. \quad (5)$$

In the next two subsections, it will be shown that how to model $\widehat{P}_{\mathrm{off}}$ and $\widehat{P}_{\mathrm{on}}$.

4.1 Dynamic off-chip memory power estimation

The dynamic power consumed in the external SRAMs is considered to be the off-chip power. Modern SRAMs possess sleep-mode where the current is much less than the operating current. In our experiments, we have used SRAM K7M323625M from Samsung as used in the Celoxica RC300 platform, where operating current and sleep current are 310 mA and 110 mA, respectively. Incorporating sleep current into static power, the dynamic off-chip memory power can be estimated as:

$$\widehat{P}_{\mathrm{off}} = V_{\mathrm{dd}} \times (I_{\mathrm{operating}} - I_{\mathrm{sleep}}) \times \frac{\#off\_m\_accesses}{\#execution\_cycles} \quad (6)$$

where $I_{\mathrm{operating}}$, I_{sleep} are the operating current and the current in the sleep-mode, respectively. $\#off\_m\_accesses$ denotes the number of accessing to the off-chip memory and $\#execution\_cycles$ is the total cycles taken by an execution of an application. They can be determined at compile time.

4.2 FPGA on-chip dynamic power estimation

To analyze composition of the FPGA on-chip dynamic

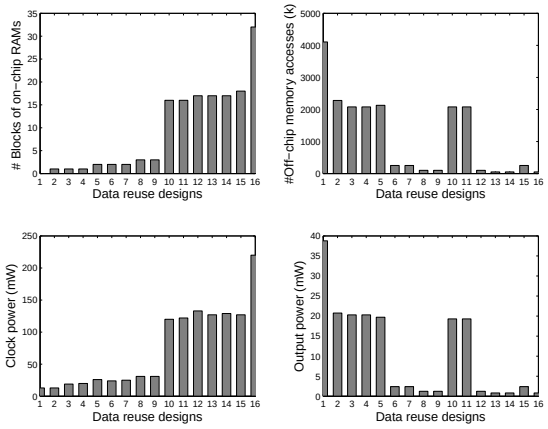

Figure 3: Experimental results of 16 different designs of the FSME algorithm.

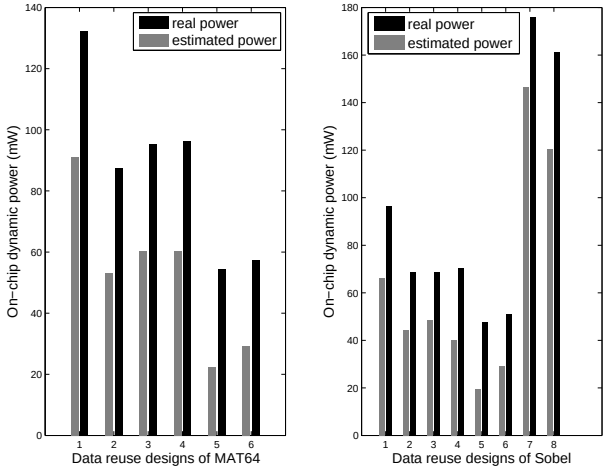

Figure 4: (a) The on-chip dynamic power of six data reuse designs of matrix-matrix multiplication and (b) The on-chip dynamic power of eight data reuse designs of Sobel detection.

power, the full search motion estimation (FSME) kernel [4] and its multiple designs with different data reuse options have been mapped on Xilinx Virtex II FPGA devices and on-chip power consumption is analyzed using Xpower. In total, 16 designs have been realized.

The on-chip dynamic power is composed of five parts, including power consumptions on input and output paths, clock nets, signal lines and logical resources. The values of each component of the on-chip dynamic power are shown in Fig. 2, except the power consumed in input paths because the values are 1 mW across all designs. Visually, the values of *clocks* and *outputs* have obvious variations over these designs, while other three components have almost constant power across the designs, because operation units and control logic of the designs are almost the same. Further investigation points out that *clocks* and *outputs* are strongly related to the memory accesses. This is illustrated in Fig. 3, where the power taken by the on-chip clocks has the same variation as the number of blocks of on-chip dedicated RAMs for different designs and the power consumed by the outputs varies as the number of off-chip memory accesses. Also, these plots show that the *clocks* and the *outputs* powers have the contrary trends for different data reuse designs, due to the fact that using larger on-chip buffers can achieve a fewer number of off-chip memory accesses.

These observations lead us to conclude that FPGA on-chip dynamic power consumption is straightforwardly related to the number of on-chip dedicated RAMs $\#BR$ and the frequency of access to the external SRAMs, $f_{\text{off\_m\_access}}$. The on-chip dynamic power P_{on} can thus be estimated by:

$$\widehat{P}_{\text{on}} = k_1 \times f_{\text{off\_m\_access}} + k_2 \times \#BR + k_3 \qquad (7)$$

where k_1, k_2, k_3 are parameters that express the on-chip power taken by accessing to off-chip memories per unit time, by a on-chip memory block and by processing units. The $\#BR$ and $f_{\text{off\_m\_access}}$ can be statically determined at compile time by analyzing the loop structure of a code after data reuse transformations. Therefore, now, only the three parameters k_1, k_2 and k_3 left unspecified.

Equation (7) is used to estimate, at compile time, the FPGA on-chip dynamic power of a design as close to the real one P_{on} as possible. For our target platform, we have found these parameters by least squares fits to data obtained from the Xpower low level power estimation tool [8]. Again, on-chip power results of the 16 designs of the FSME algorithm have been used here to obtain the parameters: $k_1 = 1.38$ mW/MHz, $k_2 = 6.88$ mW/block and $k_3 = 27.72$ mW. k_3 is algorithm-dependent, but is independent of memory accesses, *i.e.* it can be considered as a constant during data reuse exploration.

We have applied this model to data reuse designs of matrix-matrix multiplication of two 64×64 matrices and Sobel edge detection algorithm [7] to estimate the on-chip dynamic power, shown in Fig. 4. These two plots indicate that the estimated power and the real power have the similar trend and the estimated values reflect the relative merits positions of different designs. It is this relative accuracy that is used in the approach to optimize the data reuse designs.

5. EXPERIMENTAL RESULTS

We have applied our approach to solve the data reuse exploration problem for three signal and image processing kernels: full search motion estimation (FSME), Sobel edge detection (Sobel) and matrix-matrix multiplication of two 64×64 matrices (MAT64). Given a kernel, an array stored in FPGA on-chip dedicated RAMs with a single port is potentially assigned to each array reference to buffer reused data, and the array could be inserted at any level of the loop nest. The original design and data reuse designs of each kernel have been implemented in Handel-C. The luminance component of QCIF image size (144×176 pixels) is used in FSME and Sobel edge detection kernels. The platform we assumed consists of a Xilinx Virtex II FPGA and 2 banks of SRAMs. We have pipelined off-chip SRAM accesses and thus only one cycle is needed for both on-chip and off-chip memory accesses. In this way, the execution time of different designs for an algorithm only differ in loading

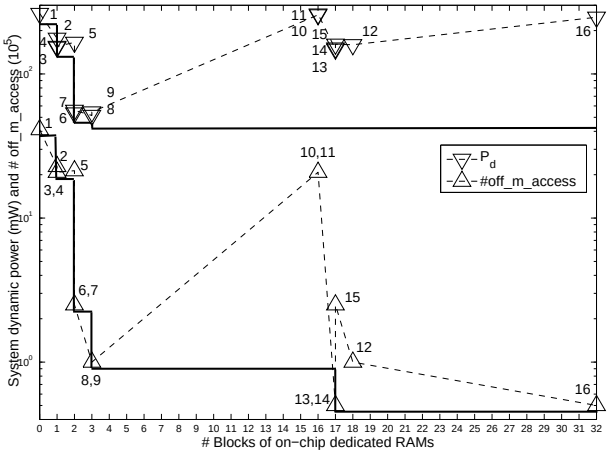

Figure 5: Results of data reuse designs of FSME, where each number corresponds to a design.

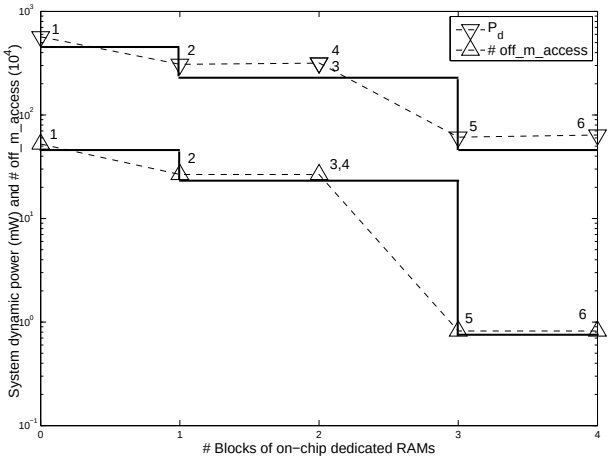

Figure 6: Results of data reuse designs of MAT64, where each number corresponds to a design.

reused data into on-chip dedicated RAMs. This difference is not large compared to the total execution time for the benchmarks. Also, all designs run at 100 MHz and the maximum frequency that each design could operate on is listed in Fig. 9. However, it is not always necessary for designs to run at the maximum frequency if the real-time application restriction is satisfied by the performance of designs. Therefore, the performance of different implementations are almost the same, meaning that energy and average power are equivalent optimization criterion.

FSME kernel operates on two sequential video frames and outputs motion vectors of the current frame against the previous frame. It has two array references (*current, previous*), corresponding to the current and previous frames respectively, accessed inside six regularly nested loops. There are three beneficial data reuse options for the reference *current*, and four for the reference *previous*. Together with the option to not have an on-chip buffer for each reference, there are

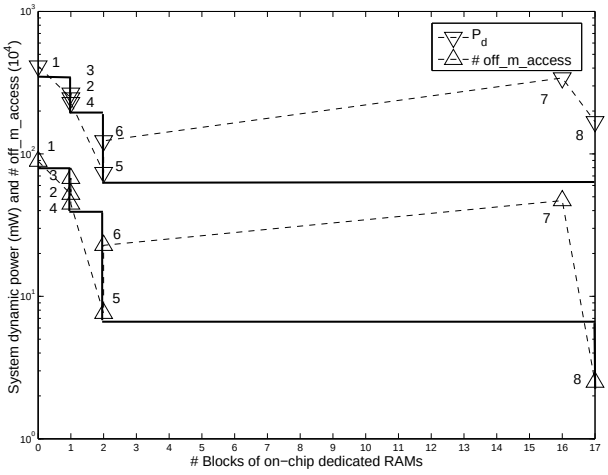

Figure 7: Results of data reuse designs of Sobel, where each number corresponds to a design.

in total 20 different options for the FSME kernel. We have implemented 16 of them and the others are ignored because they only buffer a small amount of reused data. The Sobel edge detection algorithm includes four regularly nested loops surrounding an *image* array and a *mask* array and there are in total 8 options. The MAT64 has a 3-level loop structure and two matrices under multiplication have 3 and 2 options, respectively. The options, described above, for each array reference of these kernels are listed in Table 1. Then, a data reuse design for each kernel is a combination of options of the corresponding array references. For instance, the data reuse design 1 of the FSME is the combination of OP_{11} and OP_{21}, and the data reuse design 2 is the combination of OP_{11} and OP_{24}.

The experimental results obtained by implementing all possible options of these three kernels, including system dynamic power (P_{d}), the number of off-chip memory accesses (*#off_m_access*) and the number of blocks of on-chip dedicated RAMs ($\#BR$), are plotted in Fig. 5, 6 and 7, where the y-axes are in the logarithmic scale to highlight variations of all variables in the results. It can be noticed in the figures that the system power is first decreasing as the number of blocks of on-chip dedicated RAMs is increasing and then at some point the power is starting to increase, as a result of the increase in the on-chip power consumption starts to be larger than the reduction of the off-chip power consumption from the point. Therefore, finding the optimal point is the objective of doing data reuse exploration. The most power efficient designs for each kernel over a set of specific area constraints are clearly shown and are connected using bold lines to form power efficient Pareto frontier in the figures.

An exact algorithm [14] for solving the Multiple-Choice Knapsack problem is used in the approach. Given the total number of blocks of dedicated RAMs (B) available in a FPGA device, the system dynamic power consumption (P_{ij}) estimated using the proposed power model and the required number of blocks of dedicated RAMs (B_{ij}) of every option of each array reference in the targeted kernel, this algorithm outputs the optimal solution, *i.e.* the best option for each

Table 1: The estimated #BR and dynamic power of each option of each reference of three kernels.

Kernel	Reference	Options	B_{ij}	P_{ij} (mW)
FSME	current	OP_{11}	0	119.7
		OP_{12}	16	113.6
		OP_{13}	1	8.6
		OP_{14}	1	8.6
	previous	OP_{21}	0	119.7
		OP_{22}	16	113.6
		OP_{23}	2	18.8
		OP_{24}	1	20.6
MAT64	A	OP_{11}	0	263.3
		OP_{12}	2	18.0
		OP_{13}	1	11.0
	B	OP_{21}	0	263.3
		OP_{22}	2	18.0
Sobel	image	OP_{11}	0	191.5
		OP_{12}	16	120.0
		OP_{13}	1	38.9
		OP_{14}	1	86.8
	mask	OP_{21}	0	191.5
		OP_{22}	1	7.0

reference under the area constraint.

Given B_{ij} and P_{ij} of each option of the three kernels in Table 1, for the FSME, if the area constraint B is between 3 and 32, then the algorithm gives the solution (OP_{14},OP_{23}) that is design 8, shown in Fig. 5; if B is given as 2, then the solution is (OP_{14},OP_{24}) or (OP_{13},OP_{24}) corresponding to designs 6 and 7 in Fig. 5, respectively. Here, there are two possible solutions due to the estimated P_{13} and P_{14} are the same and, in fact, designs 6 and 7 differ by only 4.2% in power. For MAT64, if B is set to 3 or 4, then the solution is (OP_{13},OP_{22}), which is design 5 in Fig. 6; if B is set to 1 or 2, then the solution is (OP_{13},OP_{21}), design 2. Similarly, for the Sobel edge detection, if B is 16, then (OP_{13},OP_{22}) is output and corresponds to design 5 in Fig. 7. Overall, it can be seen that the most power efficient design for each kernel can be obtained using the proposed approach in the context of different area constraints.

Moreover, it is worth noting that, for each kernel, the designs with the least power consumption and the designs with the least number of off-chip memory accesses under the same area constraints *may not be the same*. For FSME, in Fig. 5, if B is between 17 and 32, then the most power efficient design is design 8, while the design with the fewest off-chip memory accesses is design 14 or 16. If B is set between 3 and 16, then the most power efficient design is still design 8, while the design with the fewest off-chip memory accesses is design 8 or 9. This difference is caused by two reasons. First, more reduction of off-chip memory accesses requires more on-chip memories, leading to the more on-chip power consumption. As a result, the total system power increases. Second, designs with the same number of off-chip memory accesses may require different amounts of on-chip memories, leading to different power consumption. Thus, the power consumption is chosen as the objective of the design space exploration in the approach.

Finally, the on-chip area constraint also includes configurable logic blocks, apart from the dedicated RAMs. The number of slices used by each data reuse design of the three

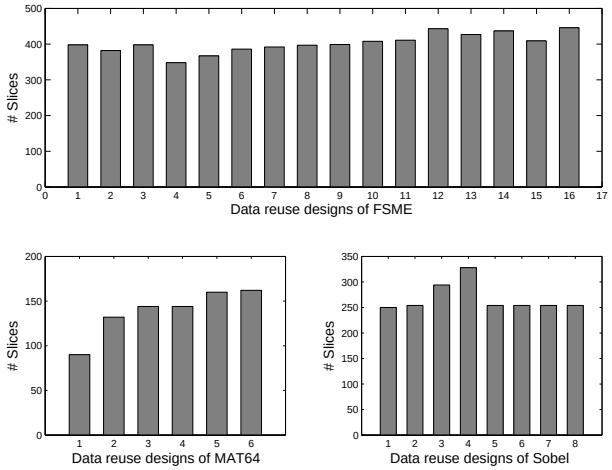

Figure 8: The amount of slices used by each design.

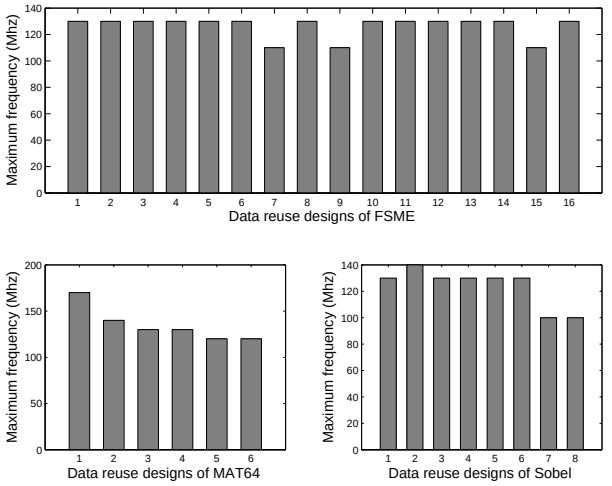

Figure 9: The maximum frequency of each design.

kernels are shown in Fig. 8. It is true that exploiting data reuse usually raises overheads in the amount of slices as well. However, it can be seen from the Fig. 8 that the variations of the amount of slices required by the designs are not large. Thus, the number of slices is not a dominated factor in the area constraint for these benchmarks.

6. CONCLUSIONS

An optimization approach for data reuse exploration with a low power objective in FPGA-based reconfigurable systems has been presented in this paper. The data reuse exploration problem under area constraints has been formulated and can be automatically solved using existing algorithms for the Multiple-Choice Knapsack problem. A dynamic power model for the reconfigurable systems is proposed with the experimental parameters. This model provides the relative power information of different data reuse

options that results in a fast and efficient design space exploration in practice. The approach has been validated by comparing the results obtained by our approach and by actually implementing experiments on several benchmarks.

In this approach, only one array stored in on-chip dedicated RAMs is assigned to each reference. In the future, we will cope with the problem where multiple arrays are introduced to buffer reused data for each reference, *i.e.* multiple reuse levels, and the arrays could be stored in on-chip dedicated RAMs or registers. Also, the multiple-port characteristic of the on-chip memory will be taken into account. Furthermore, data-level parallelization will be integrated into the data reuse exploration, and then execution time will vary significantly among different options and become an prominent optimization object.

7. REFERENCES

[1] J. Absar and F. Catthoor. Reuse analysis of indirectly indexed arrays. *ACM Trans. Des. Autom. Electron. Syst.*, 11(2):282–305, 2006.

[2] N. Baradaran and P. C. Diniz. A register allocation algorithm in the presence of scalar replacement for fine-grain configurable architectures. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 6–11, 2005.

[3] N. Baradaran, J. Park, and P. C. Diniz. Compiler reuse analysis for the mapping of data in fpgas with ram blocks. In *2004 IEEE International Conference on Field-Programmable Technology*.

[4] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[5] F. Catthoor, E. de Greef, and S. Suytack. *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[6] M. Dasygenis, N. Kroupis, K. Tatas, A. Argyriou, D. Soudris, and A. Thanailakis. Power and performance exploration of embedded systems executing multimedia kernels. *IEE Proceedings - Computers and Digital Techniques.*, 149(4):164–172, 2002.

[7] http://www.pages.drexel.edu/ ∼weg22/edge.html. accessed Aug. 2006.

[8] http://www.xilinx.com/products/design_resources /power_central. Xilinx xpower estimator user guide. accessed Jan. 2007.

[9] M. Kandemir, J. Ramanujam, M. J. Irwin, N. Vijaykrishnan, I. Kadayif, and A. Parikh. A compiler-based approach for dynamically managing scratch-pad memories in embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2):243–260, Feb. 2004.

[10] Q. Liu, G. A. Constantinides, K. Masselos, and P. Y.K.Cheung. Automatic on-chip memory minimization for data reuse. In *2007 International Conference on Field-Programmable Custom Computing Machines*, 2007.

[11] Q. Liu, K. Masselos, and G. A. Constantinides. Data reuse exploration for FPGA based platforms applied to the full search motion estimation algorithm. In *2006 International Conference on Field Programmable Logic and Applications*, pages 389–394, 2006.

[12] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[13] P. Petrov and A. Orailoglu. Performance and power effectiveness in embedded processors-customizable partitioned caches. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(11):1309–1318, Nov. 2001.

[14] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83:394–410, 1995.

[15] W.-T. Shiue, S. Udayanarayanan, and C. Chakrabarti. Data memory design and exploration for low-power embedded systems. *ACM Trans. Des. Autom. Electron. Syst.*, 6(4):553–568, 2001.

[16] D. Soudris, N. D. Zervas, A. Argyriou, M. Dasygenis, K. Tatas, C. E. Goutis, and A. Thanailakis. Data-reuse and parallel embedded architectures for low-power, real-time multimedia applications. In *PATMOS*, pages 243–254, 2000.

[17] M. Weinhardt and W. Luk. Memory access optimization for reconfigurable systems. In *IEE Proceedings on Computers and Digital Techniques*, volume 148, pages 105–112, May 2001.