Data Reuse Exploration under an On-Chip Memory Constraint for Low Power FPGA-Based 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 data dominated applications. In these applications data transfer and storage consume a large proportion of the system energy. Exploiting data reuse can introduce significant power savings, but also introduces the extra requirement for on-chip memory. To aid data reuse design exploration early during the design cycle, we present an optimization approach to achieve a power-optimal design satisfying an on-chip memory constraint in a targeted FPGA-based platform. The data reuse exploration problem is mathematically formulated and shown to be equivalent to the Multiple-Choice Knapsack Problem (MCKP). The solution to this problem for an application code corresponds to the decision of which array references are to be buffered on-chip and where loading reused data of the array references into on-chip memory happens in the code, 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 design options of an application, 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 all the benchmark circuits tested.

I. Introduction

Contemporary FPGA-based reconfigurable systems have been widely used to implement data dominated applications. This is because FPGAs with heterogeneous hardware resources successfully fill the gap between microprocessors and ASICs [1]. FPGAs could achieve higher performance and more efficient power consumption than microprocessors [2], [3]. Meanwhile, FPGAs keep higher flexibility and programmability than ASICs, although consume 9-12 times more power [4]. These factors make FPGA-based systems an attractive alternative for implementing low-volume real-time embedded systems, which require field upgrade and portability, within a shortened design cycle.

Data dominated applications, such as real-time digital signal processing, usually involve a large number of data transfers between the processor and off-chip memory, which could consume 50-80% of the embedded system energy [5]. Therefore, design space exploration aimed at reducing the proportion of power consumed by such external memory accesses is highly necessary to improve the system energy dissipation.

Data reuse, where repeatedly used data are stored in on-chip memory instead of power-hungry off-chip memory, is commonly used to reduce overall system level power [6]. An approach for exploiting data reuse in an FPGA-based platform has been proposed in [7], [8], where buffers, residing in on-chip scratch-pad memory and buffering reused data, are introduced for each array reference in a nested loop. The results show that keeping frequently used data in FPGA on-chip memories leads to a significant reduction in the off-chip memory accesses.

Clearly, the more reused data are buffered locally, the higher the reduction in off-chip memory accesses that can be achieved. At the same time, the required on-chip memory resources increase, leading to possible increases in the size of the FPGA device required and in on-chip power consumption. As a result, if the targeted FPGA is unable to provide the required on-chip memory resources, or if the increased power caused by the additional on-chip resources exceeds the power reduced by the optimization of the off-chip memory accesses, then the corresponding data reuse design is infeasible or useless. Therefore, exploration of the data reuse design space at the early design stages to minimize the system power consumption, given an FPGA on-chip memory constraint, is needed. When the target platform has multiple memory levels, and there are multiple array references accessed in a loop nest, the number of potential design options grows exponentially with the number of array references. Manual exploration of the design space is a laborious work. This paper addresses a practical approach to solve this problem automatically, which can be applied to high level synthesis of hardware.

Specifically, the main contributions of this paper are:

- a formalization of the problem of exploring data reuse space with the low power objective under an on-chip memory constraint as the well-known Multiple-Choice Knapsack Problem [9],
- a simple and experimental power consumption model for the FPGA-based reconfigurable systems, and
- an application of the proposed approach to several benchmarks, resulting in the optimal designs balancing the power consumption and the on-chip memory utilization.

The rest of the paper is organized as follows. In Section II we briefly describe related research work. Section III presents a motivational example. Section IV formulates the problem of data reuse exploration, and states the correspondence between this problem and the MCKP and an algorithm to solve the MCKP. In Section V we propose a dynamic power model. In Section VI we present experimental results from applying the approach to real benchmarks and conclude in Section VII.

II. Background

Research work aimed at achieving an optimal memory configuration has been extensively carried out in embedded system designs. The growth of the quantity of data processed in real-time multimedia and telecommunication applications means that the data memory architecture now plays a critical role in improving performance of the applications. Cache (hardware-controlled on-chip memory) is suitable to accelerate general applications, where potentially reused data are determined at run-time. In contrast, scratch-pad memory (SPM), software-controlled on-chip memory, can be configured for specific applications, where memory access patterns are known at compile-time and transfers of reused data between the off-chip memory and the SPM are scheduled statically. Therefore, for applications with regular memory accesses, such as digital signal and image processing, where the access pattern is known *a priori*, SPM is preferred [10], [11].

Extending previous work, Shiue *et al.* [5] explore data cache configurations based on cache size, execution time and energy consumption. They use the average memory access time and the memory energy consumption as metrics. This simplifies the estimates of the execution time and the energy consumption, and makes this approach feasible in the early design stages. Petrov *et al.* [12] propose a customization approach to partition the data cache and only buffer the data of array references that exhibit group 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 [6]. A cost function of power and area of the memory system is used to choose the promising memory organization. Brockmeyer *et al.* [13] refine the memory assignment and placement involved in [6] with the low energy objective. The effects of different combinations of data reuse transformations and memory system structures on the system power consumption and performance have been shown in [14] and [15] for platforms with multiple embedded instruction set processors.

Techniques have been proposed in [10] and [11] for scratch-pad memory (SPM) to analyse the data reuse existing in the nested loops and determine which level of the loop nest the data reuse is exploited in and which elements of the arrays are stored in the SPM. To make this decision, a memory access cost model, which considers the costs of accesses to the external memories and to the SPM, is used. Panda *et al.* [16] present a method for assigning data into SPM and off-chip memory with the objective to minimize cache conflicts in order to improve the speed of embedded systems. The intuitive idea of this method is that those data that are more frequently accessed and are more likely to cause cache conflicts will be stored in the on-chip SPM. The process of determining which arrays to be stored in the limited SPM space is related to the 0-1 Knapsack Problem.

The design approaches described above focus on ASICs and some of them are applicable to the FPGA-based hardware platforms. However, there are some differences between ASIC and FPGA architectures, notably the large quantity of distributed registers and the discrete sizes of on-chip RAM available on an FPGA platform. The use of on-chip embedded RAM and registers to facilitate data reuse in FPGAs has been reported in [17] and [18]. They use the following simple scheme. Given several arrays with frequently accessed data, arrays are considered individually and are entirely stored in the registers. When all register resources are exhausted, the on-chip embedded RAM is used to store the remaining arrays. When both are exhausted, accesses are made directly to the external memories. Baradaran *et al.* [19] propose an improved scheme, where the limited amount of registers are allocated over all arrays to minimize overall data access time.

Our approach, focusing on data reuse exploration in FPGA-based systems using SPM, differs from previous work described above in two main respects. Firstly, the power consumption model used in our approach is simpler than the models proposed in the previous work and is parameterizable. The simplifications, however, do not affect the accuracy when comparing between different data reuse designs for the same application, as the other factors considered by the more detailed models remain constant across all reuse design options, allowing optimization to be carried out. Secondly, on the basis of this method, we have developed an analytical formulation of the data reuse exploration problem for global optimization, obtaining the most power efficient designs. 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 algorithms for solving the MCKP can be used to find the optimal solution to our exploration problem efficiently.

III. Motivational Example

The problem under consideration is how to generate a low power FPGA-based system for a data dominated application that is implemented in imperative code, usually containing loop nests. Off-chip memory access is one of the main sources of system power consumption as discussed earlier, and thus we exploit data reuse to reduce the number of off-chip memory accesses. Buffering potentially reused data on-chip in different levels of the loop nest enables the exploration of low power designs from the imperative code.

```
/* 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)

Fig. 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.

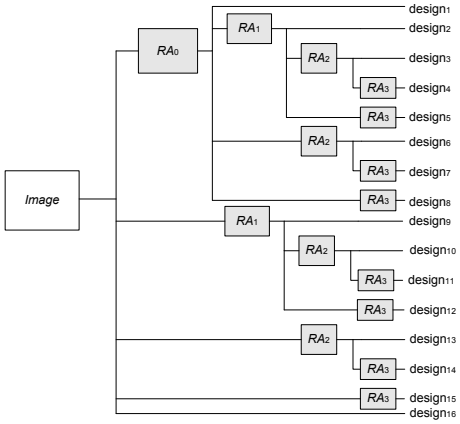

Fig. 2. Possible data reuse designs of Sobel edge detection algorithm.

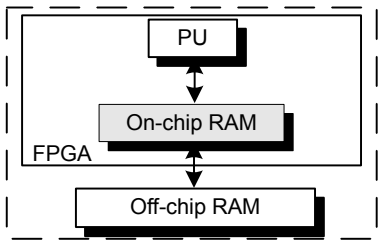

Fig. 3. The target FPGA-based platform.

This can be demonstrated using a simple example as shown in Fig. 1 (a). Following the approach in [7], in this example there are four places RA_0–RA_3, in each of which a *data reuse array* mapped to a level of the system memory hierarchy could be introduced to buffer reused data for the large array *image*. The *image* array is stored in the highest level of the memory hierarchy and the purpose of data reuse is to reduce the access count to the higher memory levels. All possible data reuse designs of this example are shown in Fig. 2, where a path from the left hand side to the right hand side is a data reuse design option and the data reuse arrays (shown in shadowed blocks) in each path are mapped onto different memory levels. Data are transferred between neighboring levels. For easy illustration, it is assumed that two memory levels are available in the target FPGA-based platform, off-chip RAM and FPGA on-chip embedded RAM, as shown in Fig. 3. Given that *image* is stored in the off-chip RAM, only designs 1, 9, 13 ,15 and 16 shown in Fig. 2 are feasible. We identify the option without data reuse and two beneficial data reuse designs[1], as shown in Fig. 1 (b), (c) and (d) corresponding to designs 16, 9 and 1 in Fig. 2. Assuming QCIF frame size for the images, the number of off-chip memory accesses of the design 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 respective requirements of 4232 bits and 202752 bits of on-chip RAM. We can see that a 9-fold reduction in the number of accesses to the off-chip memory can be obtained at the cost of 202752 bits of on-chip RAM. However, if the target FPGA contains RAMs less than this size, then the data reuse design in (d) is infeasible and the large reduction in the number of off-chip memory accesses can not be exploited.

Certainly, greater reduction in the number of off-chip memory accesses requires greater on-chip memory space, and fewer off-chip memory accesses lead to lower power consumed off-chip. However, the power models used in [6] and the power model proposed in Section V show that the power consumption of the on-chip memories increases with the size. This means that off-chip power reduction and on-chip power increase happen at the same time, resulting in uncertainty in the trend of total system power as the on-chip buffer increases.

Therefore, on one hand, exploiting data reuse to improve system power consumption is limited by on-chip memory resources

[1]A beneficial data reuse design is a design option in which for each array buffering reused data the number of accesses to this array from the next lower level is larger than from the prevous higher level.

available. On another hand, large on-chip buffers do not guarantee an optimal power solution. As a result, automatic exploration of the data reuse design space at the early design stage is very important to determine a power-optimal design, especially there exist a number of possible data reuse designs. In the next section, this problem is formulated as an Integer Linear Programming program and is automatically solved by transforming to a Multiple-Choice Knapsack Problem.

IV. Formulation of the data reuse exploration problem

The proposed data reuse approach targets the n-level loop nest surrounding multiple array references. For each array reference, there are n possible places where a data reuse array can be inserted to buffer elements of the array. A data reuse option could contain t ($1 \leq t \leq n$) data reuse arrays, each of which is present in a distinct loop level. Multiple data reuse arrays of a data reuse option are mapped on different memory levels of the target platform. Given u memory levels available in a platform, with the original large array is stored in the highest level (which has both largest size and power consumption), there could be $\sum_{t=1}^{u-1} \binom{n}{t}$ data reuse options for each reference. If m references are involved in an application, then a data reuse option of each reference has to be determined and the chosen data reuse options of all references form a data reuse design of the application. Therefore, the candidate pool has up to $(\sum_{t=1}^{u-1} \binom{n}{t})^m$ data reuse designs, which results in a very large search space for modest m. It shall be shown in this paper that choosing the best option from the large pool with the objective to minimize the system power consumption under the on-chip memory constraint can be formulated as a Multiple-Choice Knapsack Problem (MCKP).

The problem formulation described in the following targets the FPGA-based platform with two memory levels: off-chip RAM and on-chip RAM, as shown in Fig. 3, but it is easily extended to multiple memory levels. 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 has k_i options for on-chip buffer placement OP_{ij} ($1 \leq j \leq k_i$), including the option where no on-chip buffer is introduced. In each data reuse option of reference A_i there is only one data reuse array. Here, k_i may be less than $n+1$ because some data reuse options could be illusive, *i.e.* the number of reads from the buffer is not larger than the number of writes to it in the possible options [7], and k_i equal to one means that no on-chip buffers is introduced for the reference. 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 V, and occupy B_{ij} blocks of on-chip embedded RAM, a value that can be calculated following the methodology in [8]. Based on these notations, the problem can be defined as the following:

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

subject to

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

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

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

The objective function, minimization of the power consumption, is given in (1). The power consumption is memory-related as the data reuse transformation in this work only affects the data transfers and accesses between off-chip and on-chip memories. In other words, only this part of the system power consumption varies across different data reuse designs of an application. These shall be discussed in the next section in detail. Therefore, we formulate the memory-related power as the objective and the power caused by each array reference is added together. The inequality (2) defines the on-chip memory constraint, where B is the number of on-chip RAM blocks available and the sum in the left hand side is the total number of RAM blocks required to buffer data of all array references. For the case where multiple memory levels exist, inequality (2) can be applied to each level to consider the overall resource constraints. Variables x_{ij} can only be set to zero or one in (4). Here, x_{ij} is equal to one if and only if option OP_{ij} is selected for the reference A_i. 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 Problem (MCKP), and as a result algorithms used to solve the MCKP can be applied to this problem.

The Multiple-Choice Knapsack Problem is the following problem [9]. 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 data reuse options for this reference as items belonging 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 observed that our problem is, in fact, equivalent to the Multiple-Choice Knapsack Problem.

MCKP is NP-hard, but there exist several algorithms to solve the problem in pseudo-polynomial time through dynamic programming [9], [20]. The algorithm incorporated into our work is proposed in [20], which is an exact algorithm, *i.e.* the optimal solution to the MCKP is given. It starts from solving the Linear Multiple-Choice Knapsack Problem, which is obtained by relaxing the constraint $x_{ij} \in \{0, 1\}$ in the MCKP to $0 \leq x_{ij} \leq 1$, to generate an initial solution. If the solution is integer then the solution is the optimal solution to the MCKP; otherwise, a dynamic programming method is used to solve the MCKP based on the initial solution. During these two stages, the reduction phases, where two dominance criteria described in [9] are applied to delete unpromising items, are essential for effectively obtaining solutions to the problem.

We introduce this algorithm to our work to solve the problem (1)-(4). 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 embedded RAMs, B_{ij}, taken by an option is decided by the size of the data reuse array of the reference, which can be obtained by analyzing the surrounding loop structure [8]. However, an accurate power estimate for each option at the high level synthesis stage is difficult. Therefore, we focus on the development of a power model which can distinguish different data reuse design options, in order to determine which design is power-optimal. Any further accuracy is irrelevant, since the power consumption is only used as the objective function in the optimization problem.

V. Proposed power model

There exist several power models for the embedded systems. Landman's energy model has been used in [6], [14], [21], which is parameterized by the capacitances and the frequencies of read and write operations. Several power models exist for cache [5], which are based on the estimate of the cache line hit rate and several hardware-related parameters. Also some memory access models have been addressed in [10], [11], which take into account costs of transferring data between off-chip memory and SPM and of access to the SPM. Impacts of the startup of the memory access and DMA transfer on the cost are also considered. These models provide relatively accurate power estimates, but we show here that it is not necessary to model the power in such detail in order to make optimal data reuse decisions at high level synthesis stages.

For our experiments, we model the power based on the Celoxica RC300 platform [22], which is equipped with a Xilinx Virtex II FPGA and ZBT SRAMs. However, the proposed approach can be easily migrated to the platforms with recent FPGA devices once the power models are stable [23]. The model can also work with other external memories, such as DDR2 RAM, as the power consumed in the off-chip memory is assumed to increase approximately linearly with the frequency of accesses. Xpower, the low level power estimate tool [23], is used to benchmark our model. Xpower models are, in turn, constructed via low-level Spice simulations of the basic FPGA circuit elements. We follow the methodology as follows to obtain the dynamic power of FPGAs. Each design is synthesized, placed and routed targeting a Virtex II FPGA device. The HDL model of the design then is generated from the Xilinx ISE tool and has been simulated in Modelsim SE to obtain a value change dump (VCD), which contains the switching activity of all internal nets of the design. Finally, Xpower takes the placed and routed netlist and the VCD file of the design as inputs to report the power values. Following this procedure, the power accuracy estimated by Xpower is within +/-10% [23] and the power results given by Xpower vary consistently with the hardware measurements [24]. Clarke [25] also discusses the accuracy of Xpower enough to enable the building of high-level power models. We believe that this "bootstrapping" of higher-level power models upon lower-levels is an appropriate experimental methodology. Therefore, in the remaining of this paper, we will verify our model based on the Xpower power reports.

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

$$\widehat{P}_{\mathrm{d}} = \widehat{P}_{\mathrm{off}} + \widehat{P}_{\mathrm{on}}. \tag{5}$$

where $\widehat{P}$ is used to express the estimated value of power P. In the next two subsections, it will be shown how $\widehat{P}_{\mathrm{off}}$ and $\widehat{P}_{\mathrm{on}}$ are modeled.

A. Dynamic off-chip power estimate

In this paper, external SRAM is considered as the off-chip memory. The dynamic power consumed in the external SRAM is considered to be the off-chip power. The off-chip SRAM accesses are pipelined and thus one cycle is needed for an off-chip access. 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. Its operating current and sleep current are 310 mA and 110 mA, respectively [26]. We assume that dynamic power consumed on the off-chip SRAM appears only when the SRAM is being accessed. Incorporating the sleep current into the static power, the dynamic off-chip memory power can be estimated as:

$$\widehat{P}_{\mathrm{off}} = V_{\mathrm{dd}} \times (I_{\mathrm{operating}} - I_{\mathrm{sleep}}) \times f_{\mathrm{off\_m\_access}} \tag{6}$$

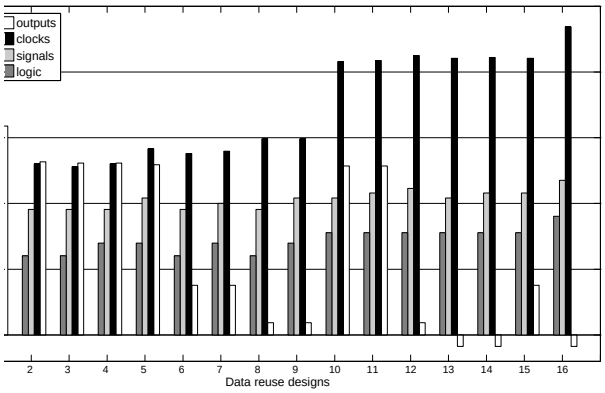

Fig. 4. The composition of the on-chip dynamic power of the FSME implementations.

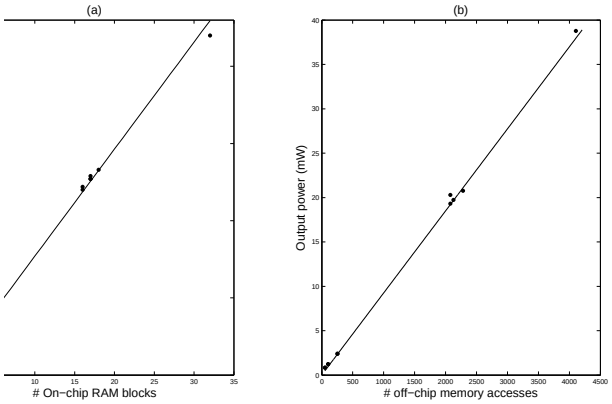

Fig. 5. Correlation analysis of the results of the FSME algorithm. (a) The linear correlation between the number of RAM blocks used and the power consumed in the *clocks*. (b) The linear correlation between the number of off-chip memory accesses and the power consumed in the *outputs*.

where $I_{\text{operating}}$ and I_{sleep} are the operating current and the sleep-mode current of the SRAM, and $f_{\text{off\_m\_access}}$ is the frequency of accesses to the external SRAM. The access frequency is determined at compile time, since the number of accesses to the external SRAM and the total execution time of a design can be counted statically.

B. FPGA on-chip dynamic power estimate

To analyze composition of the FPGA on-chip dynamic power, the full search motion estimation (FSME) kernel [27] 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 the power consumed on input paths, output paths, clock nets, signal lines and logical resources. The value of each component of the on-chip dynamic power for each design of the FSME algorithm is shown in Fig. 4, where each group of bars shows the power components of a design, except for the power consumed in the 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 two components have almost constant power across the designs, because computation units and control logic in the different data reuse designs are almost the same. Further investigation points out that *clocks* and *outputs* are strongly related to the memory access. This is illustrated in Fig. 5, where each dot corresponds to a design of the FSME algorithm. It is clear shown that in Fig. 5 (a) the power taken by the clock nets is linearly correlated with the number of on-chip RAM blocks required by different designs and in Fig. 5 (b) the power consumed by the output paths is linearly correlated with the number of off-chip memory accesses. In both cases, the correlation coefficients are more than 0.999 and the corresponding p-values are negligible.

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

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

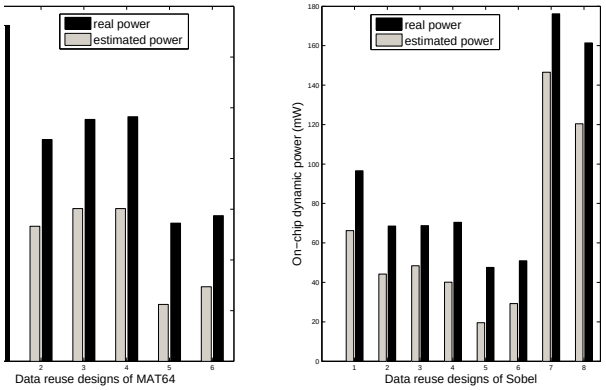

Fig. 6. (a) The on-chip dynamic power of six data reuse designs of matrix-matrix multiplication and (b) The on-chip dynamic power of eight data reuse designs of Sobel detection.

where k_1, k_2, k_3 are constants that express the on-chip power taken by accessing to the off-chip memories per unit time, by an on-chip memory block per unit time and by computation units and control logic. k_3 is algorithm-dependent, but is independent of the memory access, *i.e.* the power consumed on the computation units and control logic can be considered as a constant during the data reuse exploration of an application, since the data reuse transformation does not change the computation operations. The parameters $\#BR$ can be statically determined at compile time by analyzing the loop structure of a code after data reuse transformations. The constants k_1 and k_2 can be estimated experimentally once for each target platform; in this paper we do so for the Celoxica RC300 board. Then, we use the constants obtained for the applications implemented on the same platform.

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 the target platform, we have found the constants k_1, k_2 and k_3 by least-squares fits to data obtained from Xpower [23]. Again, on-chip power results of the 16 designs of the FSME algorithm have been used here to obtain the constants: $k_1 = 1.38$ mW/MHz and $k_2 = 0.07$ mW/MHz. Note that the parameter k_1 and k_2 represent the power per frequency unit allowing the model to be used at different frequencies. As k_3 can be seen as a constant across different reuse options of a design, we use the following model for on-chip dynamic power estimate, which is the memory-related power.

$$\widehat{P}_{\text{on}} = 1.38 \times f_{\text{off\_m\_access}} + 0.07 \times \#BR \quad \text{mW}. \tag{8}$$

We have applied the model (8) to data reuse designs of matrix-matrix multiplication of two 64×64 matrices (MAT64) and Sobel edge detection (Sobel) algorithm [28] to estimate the on-chip dynamic power, shown in Fig. 6. The number of on-chip RAM blocks $\#BR$ and the frequency of accesses to the external SRAMs, $f_{\text{off\_m\_access}}$ are statically determined for each design of these two benchmarks before translating the algorithms to RTL files. The real dynamic power values shown in this figure are obtained using Xpower after the place and route process for each design. These two plots indicate that the estimated power and the real power have the similar trend and the estimated values reflect the relative merit positions of different designs, although the estimated values are smaller than the real one due to disregarding k_3. It is this *relative* accuracy that is used in the approach to choose the data reuse designs. These results, also, show that the power model obtained by means of the FSME algorithm validates on these two algorithms. In this way, the proposed power model is considered to be consistent and thus effective on distinguishing data reuse designs of each algorithm. It is not the absolute accuracy that is of interest to us.

VI. 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, a data reuse array stored in FPGA on-chip embedded RAMs is potentially assigned to each array reference to buffer reused data, and the data reuse 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 a C-like hardware description language, Handel-C [29]. The luminance component of QCIF image size (144×176 pixels) is used in the FSME and Sobel kernels. The platform we have used consists of a Xilinx Virtex II FPGA and 2 banks of SRAM. 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 reused data into on-chip embedded RAMs. This difference is not large compared to the total execution time for the benchmarks, under 5% in most cases. Also, targeting a real-time environment, all designs are fixed to run at 100 MHz, although the maximum frequency that each design could operate on is listed in Fig. 7. Therefore, the

TABLE I
THE DETAILS OF THREE KERNELS.

Kernel	Reference	Options	B_{ij}	$\widehat{P}_{ij}$ (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

execution times of different implementations are almost the same, meaning that energy and average power are the equivalent optimization criterion. The Handel-C descriptions of the designs have been synthesized using Celoxica's DK Design Suite and mapped on devices using Xilinx ISE software. After place and route, each design is simulated using Modelsim and then total on-chip power consumption is analyzed using Xpower.

The details of three benchmarks are shown in Table I. The 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. We consider three beneficial data reuse options $\{OP_{i2}, OP_{i3}, OP_{i4}\}$ and the option to not have an on-chip buffer OP_{i1} for each reference, *i.e.* data are accessed directly from the off-chip memories. A data reuse design of the kernel is a combination of the reuse options of the array references involved in it. Thus there are in total 16 different data reuse designs for the FSME kernel. For instance, data reuse design 1 of the FSME in Fig. 4 is the combination of OP_{11} and OP_{21}, and data reuse design 2 is the combination of OP_{11} and OP_{24}. The MAT64 has a 3-level loop structure and two matrices under multiplication have 3 and 2 reuse options, respectively. Therefore, 6 data reuse designs are considered. The Sobel edge detection algorithm includes four regularly nested loops surrounding an *image* array and a *mask* array and there are in total 8 design options, as one array reference has 4 reuse options and another has 2. The number of on-chip RAM blocks B_{ij} required by a data reuse option to buffer reused data of each array reference are obtained by our previous work [8] and are shown in the table. In the last column of Table I, the dynamic power consumption of each data reuse option is estimated by the power model (8) proposed in Section V. The system dynamic power consumption of a design for each kernel is considered in this paper as a sum of dynamic power of chosen data reuse options of all array references of the kernel.

An exact algorithm [20] for solving the Multiple-Choice Knapsack Problem is used in the approach. Given the total number of RAM blocks (B) available in an FPGA device, B_{ij} and $\widehat{P}_{ij}$ of all reuse options of each array reference in the target kernel, the problem (1)–(4) is solved by this algorithm and a solution corresponding to the power-efficient design is output.

In Figs. 8 (a), 9 (a) and 10 (a), for a range of on-chip RAM constraints, the designs proposed by our approach are shown in downward-pointing triangles with numbers, which are used to distinguish different data reuse designs of a kernel, and connected using bold lines to form power-efficient Pareto frontiers. In some cases several numbers share the same triangle symbol, because the estimated system dynamic power consumption of the corresponding data reuse designs is the same. For the FSME case in Fig. 8 (a), if the on-chip memory constraint B is larger than 2 blocks, then the approach proposes the solution (OP_{14},OP_{23})—design 8—as the power-efficient design; 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, respectively, and either can be chosen as the power-efficient design under this memory constraint. Here, there are two possible solutions due to the estimated power P_{13} and P_{14} being equal as shown in Table I and, in fact, designs 6 and 7 differ by only 4.2% in power once implemented. For MAT64 in Fig. 9 (a), if B is larger than 2, then the proposed power-efficient design is (OP_{13},OP_{22}), which is design 5; if B is set to 1 or 2, design 2 (OP_{13},OP_{21}) is proposed. Similarly, for the Sobel edge detection in Fig. 10 (a), if B is 16, then (OP_{13},OP_{22}) is output and corresponds to

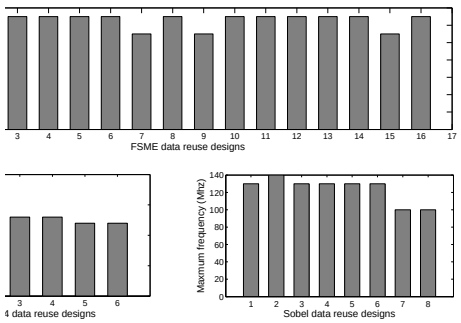

Fig. 7. The maximum frequency of each design.

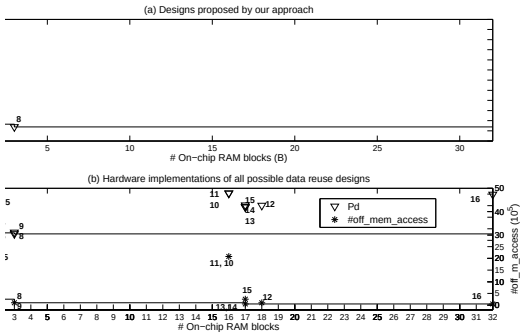

Fig. 8. Experimental results of FSME, where each number corresponds to a design. (a) Power-optimal designs proposed by our approach. (b) Implementation of all possible data reuse designs.

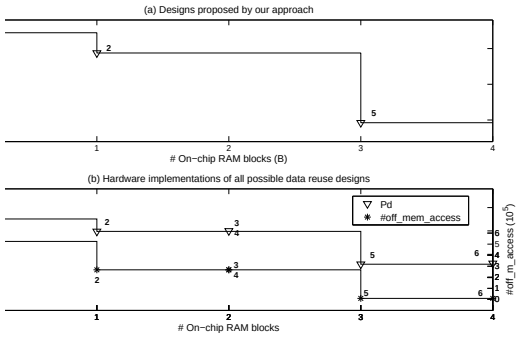

Fig. 9. Experimental results of MAT64, where each number corresponds to a design. (a) Power-optimal designs proposed by our approach. (b) Implementation of all possible data reuse designs.

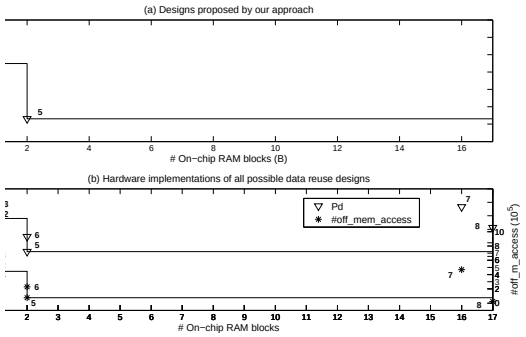

Fig. 10. Experimental results of Sobel, where each number corresponds to a design. (a) Power-optimal designs proposed by our approach. (b) Implementation of all possible data reuse designs.

design 5. Each of these power-efficient designs is generated within a second by solving the problem (1)–(4).

To verify the correctness of the proposed approach to determine the power-efficient designs, possible data reuse designs of three kernels, as listed in Table I, have been implemented on the target platform after synthesis, placement and routing. The experimental results obtained are plotted in Figs. 8 (b), 9 (b) and 10 (b), where the different designs are numbered as well. Each figure shows two groups of results, system dynamic power (P_{d} shown in downward-pointing triangles) and the number of off-chip memory accesses (*#off_m_access* shown in asterisks) of different reuse designs with the corresponding on-chip RAM block requirement. The most power-efficient designs for each kernel form the power-efficient Pareto frontier in the upper part of each figure. By comparing the power-efficient Pareto frontier in subfigure (a) with one obtained by implementing all possible designs on an FPGA device in (b) in Figs. 8, 9 and 10, it can be seen that the most power-efficient designs for each kernel can be obtained by the proposed approach in the context of different on-chip memory constraints. This also verifies the power model proposed in Section V. Moreover, it can be noticed that the system dynamic power is not monotonically decreasing with the number of on-chip RAM blocks available for buffering data, but at some point the power starts to increase, as the increase in the on-chip power consumption starts to be larger than the reduction in the off-chip power consumption. This demonstrates the necessity of doing data reuse exploration to find the most power-efficient designs.

Moreover, the comparison of two groups of results in Figs. 8 (b), 9 (b) and 10 (b) shows 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 memory constraints *may not be the same*. For the FSME case, in Fig. 8 (b), if the number of RAM blocks available on-chip 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 the number of RAM blocks is 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, the more reductions in the off-chip memory accesses (by buffering data) requires the more on-chip memories, leading to the more on-chip power consumption. When the increase in on-chip power is greater than the reduction in off-chip power, the total system power increases. In this case the on-chip power overhead involved means that it is not worth buffering these data, and this is correctly captured by our approach. 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 correctly captured as the objective of the design space exploration in the approach.

VII. Conclusion

An optimization approach for data reuse exploration with the low power objective in the FPGA-based systems has been presented in this paper. The data reuse exploration problem under the on-chip memory constraint has been formulated as the Multiple-Choice Knapsack Problem and can be automatically solved using existing algorithms. 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 leads to a fast and efficient design space exploration in practice. The approach has been validated by comparing the results obtained by the proposed approach and by actually implementing experiments on several benchmarks.

In this approach, the dependence of program parallelization and memory system design is not explored. In the future, 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.

References

[1] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, 2002.

[2] Z. Guo, W. Najjar, F. Vahid, and K. Vissers, "A quantitative analysis of the speedup factors of FPGAs over processors," in *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2004, pp. 162–170.

[3] V. Bonato, E. Marques, and G. A. Constantinides, "A floating-point extended kalman filter implementation for autonomous mobile robots," *VLSI Signal Processing*, to be published.

[4] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *FPGA '06: Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2006, pp. 21–30.

[5] W.-T. Shiue, S. Udayanarayanan, and C. Chakrabarti, "Data memory design and exploration for low-power embedded systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 6, no. 4, pp. 553–568, 2001.

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

[7] 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*, 2006, pp. 389–394.

[8] Q. Liu, G. A. Constantinides, K. Masselos, and P. Y. K. Cheung, "Automatic on-chip memory minimization for data reuse," in *FCCM '07: Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 251–260.

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

[10] J. Absar and F. Catthoor, "Reuse analysis of indirectly indexed arrays," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 2, pp. 282–305, 2006.

[11] 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*, vol. 23, no. 2, pp. 243–260, Feb. 2004.

[12] 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*, vol. 20, no. 11, pp. 1309–1318, Nov. 2001.

[13] E. Brockmeyer, M. Miranda, H. Corporaal, and F. Catthoor, "Layer assignment techniques for low energy in multi-layered memory organisations," in *Proc. 6th ACM/IEEE Design, Automation and Test in Europe Conference and Exhibition*, Munich, Germany, Mar. 2003, pp. 1070–1075.

[14] M. Dasygenis, N. Kroupis, K. Tatas, A. Argyriou, D. Soudris, and A. Thanailakis, "Power and performance exploration of embedded systems executing multimedia kernels," *IEE Proc.-Comput. Digit. Tech.*, vol. 149, no. 4, pp. 164–172, 2002.

[15] 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 '00: Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation*. London, UK: Springer-Verlag, 2000, pp. 243–254.

[16] P. R. Panda, N. D. Dutt, and A. Nicolau, "On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 682–704, 2000.

[17] 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*, Dec. 2004, pp. 145–152.

[18] M. Weinhardt and W. Luk, "Memory access optimization for reconfigurable systems," in *IEE Proceedings on Computers and Digital Techniques*, vol. 148, May 2001, pp. 105–112.

[19] 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*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 6–11.

[20] D. Pisinger, "A minimal algorithm for the Multiple-Choice Knapsack Problem," *European Journal of Operational Research*, vol. 83, pp. 394–410, 1995.

[21] 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*, 2000, pp. 243–254.

[22] Celoxica, "RC300 board specifications," accessed Jan. 2007. [Online]. Available: http://www.celoxica.com/techlib/files/cel–w040216143f–257.pdf

[23] Xilinx, "Xilinx Xpower estimator user guide," accessed Jan. 2007. [Online]. Available: http://www.xilinx.com/products/design_resources/power_central

[24] L. Wang, M. French, A. Davoodi, and D. Agarwal, "FPGA dynamic power minimization through placement and routing constraints," *EURASIP J. Embedded Syst.*, vol. 2006, no. 1, pp. 7–7, 2006.

[25] J. A. Clarke, "High-level power optimization for digital signal processing in reconfigurable logic," Ph.D. dissertation, Imperial College London, London, UK, 2008.

[26] http://datasheet.digchip.com, "1Mx36 & 2Mx18 flow-through NtRAM datasheet," accessed Jan. 2007.

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

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

[29] http://www.celoxica.com, "Handel-C language reference manual," accessed Aug. 2006.