Exploiting Reconfigurable SWP Operators for Multimedia Applications

Daniel Menard, Nguyen Hai Nam, François Charot, Stéphane Guyetant, Jérémie Guillot, Erwan Raffin, Emmanuel Casseau

EXPLOITING RECONFIGURABLE SWP OPERATORS FOR MULTIMEDIA APPLICATIONS

D. Menard [1], *H.N. Nguyen* [1], *F. Charot* [1], *S. Guyetant* [2], *J. Guillot* [3], *E. Raffin*[1] , *E. Casseau* [1]

[1] IRISA/INIRA, Campus de Beaulieu, F-35000 Rennes, Email: name@irisa.fr
[2] CEA, LIST, CEA/Saclay F-91191 Gif-Sur-Yvette, Email: stephane.guyetant@cea.fr
[3] LIRMM, 161 rue Ada, F-34095 Montpellier, Email: jeremie.guillot@lirmm.fr

ABSTRACT

Implementing image processing applications in embedded systems is a difficult challenge due to the drastic constraints in terms of cost, energy consumption and real time execution. Reconfigurable architectures are good candidates to take-up this challenge and especially when the architecture is able to support different word-lengths of pixel through Sub-Word Parallelism (SWP) capabilities. Exploiting the diversity of supported data-types requires automation tools able to optimize the data word-length under an accuracy constraint. In this paper, a new approach for word-length optimization in the case of SWP operations is proposed. Compared to existing approaches the optimization time is significantly reduced without sacrificing the quality of the optimized solution. The results show the ability of our approach to exploit the SWP capabilities associated with multimedia processors.

1. INTRODUCTION

Multimedia applications are more and more popular in embedded systems. Since several years, nomadic objects integrate real-time video and image processing applications. Implementing these applications in embedded systems is a difficult challenge due to the drastic constraints in terms of cost, energy consumption and real time execution. Image processing at pixel level, like image filtering, edge detection, pixel correlation or at block level such as motion estimation have to be considered. Such applications are typically computationally intensive and require energy efficient architectures. Moreover, the flexibility of the architecture is required to adapt to the diversity of the processing patterns associated to the different standards and the diversity of the data types handled by these applications.

In the ROMA project [7, 9], a reconfigurable processor able to adapt its computing structure to image processing applications has been developed. To avoid complex interconnection networks between operators, the processor is made-up of a pipeline of flexible coarse-grain reconfigurable operators exhibiting efficient power and performance features. The architecture provides the reconfigurability at the data word-length level by supporting different word-lengths of data and at the operation level by supporting different multimedia oriented operations. This processor provides a trade-off between computation accuracy and implementation cost through SWP capabilities.

This kind of architectures exhibits high performance but design automation tools are required to bridge the gap between more and more complex applications and high performance hardware platforms and thus, to reduce the time-to-market. As mentioned in [2], one of the most time consuming part of the implementation process is the fixed-point conversion and especially the word-length optimization stage. Most of the works on word-length optimization focusses on the case of hardware implementation [1]. In this case, the architecture is defined by the developer and the operator word-length can take any value in a given range. In the case of our architecture only several word-lengths are available for each operation due to the availability of SWP capabilities. The methodologies presented in [6, 4] target fixed-point processors and carry out a floating-point to fixed-point transformation leading to an ANSI-C code with integer data types. Nevertheless, the different data types supported by the architecture are not taken into account. In [8], the data word-lengths are optimized in the case of architecture with SWP operators. The optimization process is based on a tree model and a branch and bound algorithm is used to find the best solution. This approach is suitable for applications having a limited number of variables to optimize or when the number of supported data types is limited. For our ROMA architecture, five different data types, suitable for multimedia applications, are supported.

In this paper, a new approach for data word-length optimization is proposed for the case of architectures supporting several data types. Our approach is based on two stages. First an initial optimized solution is found with a heuristic and then this solution is refined to improve the quality of the solution. This approach allows obtaining an optimized solution with a significant reduction of the optimization time compared to existing approaches. The results show the ability of our approach to exploit efficiently the SWP capabilities associated with multimedia processors.

The paper is organized as follow. In Section 2, the main characteristics of our reconfigurable architecture are summarized and the flexible operator is presented. The fixed-point conversion process is described in Section 3 and especially, the word-length optimization problem is presented. In Section 4, the proposed optimization approach is detailed. The efficiency of our approach is illustrated through different experiments in Section 5. Finally, Section 6 draws conclusions.

2. FLEXIBLE RECONFIGURABLE ARCHITECTURE

The ROMA processor is a reconfigurable datapath of operators, controlled by a low footprint sequencer that trigs the execution and the loading of configurations. The datapath part is composed of a set of local scratchpad memories, each associated to an address generator that can create streams of data, a set of reconfigurable operators and an optimized interconnect. The latter creates computing patterns, with data read from local memories, processed through a pipeline of operators and written back in selected memories. The controller handle streams over bursts of data and can as well tune the datapath

The work presented in this paper is supported by the French Architectures du Futur ANR program ANR-06-ARFU-004.

at each clock cycle. For more details on the architecture, the reader can refer to [7].

For performance enhancement, reconfigurable processors have to overcome the overheads of reconfigurations such as complexity of interconnection network and reconfiguration time. In processors dealing with multimedia applications these overheads can be reduced by providing the reconfigurability inside the processing units rather than at the interconnection level. This reconfigurable operator is made-up of two inputs x and y and one output z. This operator is able to carry-out accumulation on different patterns. These patterns can be addition/substraction ($x_i \pm y_i$), multiplication ($x_i \times y_i$), absolute value of difference ($|x_i - y_i|$). Signed and unsigned operations can be carried-out. The operator input and output word-length is equal to 40 bits.

Due to low precision data nature of multimedia applications, reconfiguration at the operator level also provides additional speed-up through the exploitation of data-level parallelism. Our processor provides Sub-Word Parallelism (SWP) capabilities. An operator of word-length N is split-up to execute k operations in parallel on sub-words of word-length N/k. This technique can accelerate the code execution time up to a factor k. Actually, the SWP operator can compute five 8-bit operations or four 10-bit operations or three 12-bit operations or two 16-bit operations or one 40-bit operation. This operator not only eliminates the need of reconfiguration time but also provides the reconfigurability at both data size level (different pixel data sizes) and at operation level (different multimedia oriented operations). This ensures better a utilization of processor's resources and reduces the reconfiguration overheads significantly.

3. DESIGN FLOW

The general flow to map an application described with a C code with floating data-types to the ROMA architecture is based on the generic compilation infrastructure *GECOS*. Firstly, the fixed-point conversion of the application is carried-out. The data word-length is optimized according to the data-types supported by our architecture. A Control Data Flow Graph (CDFG) is used as intermediate representation. Then, the fixed-point application is mapped to the reconfigurable architecture with the DURASE flow [9].

3.1. Fixed-point conversion

The fixed-point conversion process defines the data type associated with each variable. This conversion process can be divided into two main modules. The first part corresponds to the determination of the integer part word-length of each data. The number of bits for this integer part must allow the representation of all the values taken by the data, and is obtained from the data bound values. This first part requires to evaluate the dynamic range of each data. The interval arithmetic technique is used for this evaluation.

The second part corresponds to the determination of the fractional part word-length which depends on the data types supported by the processor. The data types which minimize the implementation cost and respect the accuracy constraint are selected. This process is detailed in the next section.

3.2. Data word-length optimization

The data word-length optimization process must explore the diversity of the data types supported by the processor. The goal of this step is to minimize the implementation cost C under a given accuracy constraint $P_{b_{max}}$. The fixed-point accuracy is evaluated through the quantization noise power P_b. Let $\mathbf{wl}$ be the word-length of the different operations. The optimization problem is modelled as follows

$$\min\left(C(\mathbf{wl})\right) \quad \text{subject to} \quad P_b(\mathbf{wl}) < P_{b_{max}} \tag{1}$$

Our methodology selects the SWP configuration which respects the global accuracy constraint $P_{b_{max}}$ and minimizes the implementation cost. For the optimization process, the cost $C()$ (implementation cost) and the constraint $P_b()$ (fixed-point accuracy) functions must be evaluated.

3.2.1. Constraint evaluation

Most of the available approaches to evaluate the computation accuracy due to fixed-point arithmetic are based on a bit-true simulation of the fixed-point application [5] [3]. Nevertheless, this technique suffers from a major drawback which is the time required for the simulations. An alternative to the simulation based methods is an analytical approach. The main advantage is the reduction of the execution time for the fixed-point optimization process. Indeed, the accuracy metric expression is determined only once, then, the fixed-point system accuracy is evaluated through the computation of a mathematical expression. The method presented in [10] is used to obtain this analytical expression.

3.2.2. Cost evaluation

The aim of this module is to estimate the global implementation cost according to the SWP configuration selected for each CDFG operation. Nevertheless, the goal is not to obtain an exact estimation of the implementation cost but to compare and to select between several SWP configurations. Thus, a simple estimation model is used to evaluate the application implementation cost C. This cost can be the execution time or the energy consumption. The execution time is considered in the rest of the paper.

An operation o_i is characterized by its function γ_i. The number of times, n_i, that this operation is executed is determined statically for each operation o_i. Let $m(\mathbf{wl_i}, \gamma_i)$ be the number of operations of type γ_i which can be executed in parallel on the reconfigurable operator when the operand word-length is equal to $\mathbf{wl_i}$. Let $c(\gamma_i)$, be the cost to execute an operation of type γ_i. Let N_o, be the number of operations in the CDFG. The expression of the global implementation cost C according to the operation word-length $\mathbf{wl}$ is as follow

$$C(\mathbf{wl}) = \sum_{i=1}^{N_o} \left\lceil \frac{n_i}{m(\mathbf{wl_i}, \gamma_i)} \right\rceil c(\gamma_i) \tag{2}$$

4. OPTIMIZATION ALGORITHM

The data word-length optimization is achieved in two stages. First, an initial optimized solution is obtained with a heuristic corresponding to an extension, for SWP operators, of the approach presented in [1]. This heuristic allows obtaining an initial optimized solution quickly but does not always select the optimal solution. Thus, this optimized solution is refined with a *branch & bound* algorithm integrating different techniques to reduce drastically the search space.

4.1. Determination of the initial optimized solution

For each operation o_i, the list of the N_{SWP} SWP configurations which can be used are determined. Let $\mathbf{wl^i_{SWP}}$ be the vector

storing in rising order the word-length of the operand associated with each SWP configuration. Let $\mathbf{WL}$ be a $N_o \times N_{SWP}$ matrix representing the search space for the optimization problem. This matrix $\mathbf{WL}$ groups together the vector $\mathbf{wl^i_{SWP}}$ as follows $\mathbf{WL}(i, 1 \dots N_{SWP}) = \mathbf{wl^i_{SWP}}'$.

The proposed heuristic, described in Algorithm 1 is made-up of two steps corresponding to the determination of a starting solution and to the search of the optimized solution which allows respecting the accuracy constraint. The starting solution $\mathbf{WL}(\mathbf{j_{mwc}})$ corresponds to the combination of the minimal word-length associated to each operation. The minimal word-length for operation o_i corresponds to the word-length which minimizes the cost and fulfil the accuracy constraint when all the other word-lengths are set to their maximal value. For the search of the optimized solution $\mathbf{WL}(\mathbf{j_{min}})$, an iterative approach is used and the combination of the minimal word-length is used as starting solution. In this case, the accuracy constraint is no longer fulfilled. At each iteration, the SWP configuration associated with one operation o_i is changed. The choice of this operation is done from the computation of the metric ∇ which evaluates the gain in terms of cost and accuracy for this new SWP configuration. The metric ∇_i computes the ratio between the gradient of the accuracy and the gradient of the cost for operation o_i

$$\nabla_i \leftarrow \frac{Pb(\mathbf{wl_\Delta}) - Pb(\mathbf{wl})}{C(\mathbf{wl_\Delta}) - C(\mathbf{wl})} \tag{3}$$

The operation which minimizes the cost increase and maximizes the accuracy increase is chosen. The algorithm stops when the accuracy constraint is fulfilled.

Algorithm 1 WL Optimization for SWP processors

```
{ — Starting solution search —}
for i = 0 ... N_o do
    wl ← WL(1 ... N_o, N_SWP)
    j = N_SWP
    while P_b(wl) > Pb_max do
        j ← j − 1
        wl ← WL(1 ... N_o, j)
    end while
    j_mwc(i) = j + 1
end for
{ — Optimized solution search —}.
∀ i = 0 ... N_o   wl(i) ← WL(i, j_mwc(i))
j = j_min
while P_b(wl) < Pb_max do
    for i = 0 ... N_o do
        wl_Δ ← wl
        wl_Δ(i) ← WL(i, j(i) + 1)
        ∇_i ← (Pb(wl_Δ) − Pb(wl)) / (C(wl_Δ) − C(wl))
    end for
    k ← argmax_i(∇_i)
    j(k) ← j(k) + 1
    wl(k) ← WL(k, J(k))
end while
j_min = j
```

4.2. Refinement of the initial optimized solution

To refine the optimized solution $\mathbf{WL}(\mathbf{j_{min}})$ obtained with the heuristic presented above, a *branch & bound* (*B&B*) algorithm is carried out on a limited search space to obtain reasonable optimization times. Different techniques are used to limit drastically the search space. The search space retained for this *B&B* algorithm is made-up for each operation o_i of the SWP configuration $\mathbf{j_{min}}(i)$ obtained with the heuristic and the previous $(\mathbf{j_{min}}(i) - 1)$ and next $(\mathbf{j_{min}}(i) + 1)$ SWP configurations of the initial search space. Thus, each optimization variable has a maximal number of three values. This limitation to three values per variable is applied because the probability that the distance between the optimal solution and the initial optimized solution $\mathbf{j_{min}}$ is greater than two SWP configurations is very low. Indeed, in this case, for our ROMA architecture, it implies a difference of more than 4 bits between the two solutions. Let $\mathbf{WL}(\mathbf{j_{opt}})$ be the solution obtained with the refinement approach.

This optimization technique based on a tree exploration is node-evaluation-order sensitive. To find quickly a good solution in order to reduce the search space, the variables with the greatest influence on the optimization process must be evaluated first. The metric ∇_i is used to order the optimization variables. The variables are ordered by decreasing values of ∇_i.

In the *B&B* algorithm, the partial solutions are evaluated to stop the tree exploration, if they can not lead to the best solution. At the tree level l, the exploration of the subtree induced by the node representing $\mathbf{wl(l)}$ can be stopped if the minimal execution time which can be obtained during the exploration of this subtree is greater than the minimal execution time which has already been obtained. The minimal execution time is determined by selecting for operation o_j $(j \in [l+1, N_o])$, the SWP configuration with the minimal cost. At the beginning of the *B&B* algorithm, the minimal cost is initialized with the optimized cost $C(\mathbf{WL}(\mathbf{j_{min}}))$ obtained with the heuristic algorithm. Compared to a classical *B&B* algorithm, it is not necessary to find a first solution to be able to apply this search space limitation technique. This initial value for the minimal cost allows avoiding the exploration of many branches of the tree.

Likewise, at the tree level l, the exploration of the subtree induced by the node representing $\mathbf{wl(l)}$ can be stopped if the minimal value of the quantization noise power, which can be obtained during the exploration of this subtree, is higher than the precision constraint ($P_{b_{max}}$). The SQNR maximal value is obtained by fixing the word-lengths $\mathbf{wl(j)}$ $(j \in [l+1, N_o])$ to their maximal values.

5. EXPERIMENTS

Different experiments have been carried-out on multimedia kernels to underline the efficiency of our approach. The assumption that the data stored in the local memory are aligned is used. The Pareto curve for the Sum of Absolute Difference algorithm is given in figure 1. The Pareto curve gives the best implementation cost obtained for different accuracy constraints and shows the trade-off between the computation accuracy and the implementation cost. The number of variables in the optimization process is limited because all operations processing the different elements of a same vector use the same fixed-point format. This constraint for the word-length optimization allows obtaining a more homogeneous solution able to exploit the SWP capabilities of the architecture. On the Pareto curve, four points s_x are plotted and correspond to the solution obtained for a SWP configuration using x-bit input. For this algorithm, the number of points s_x in the Pareto curve is limited because, the reconfigurable operator is able to implement directly the pattern associated to the SAD algorithm and thus, the SWP configuration fixes the word-lengths inside the SAD pattern. The implementation cost corresponding to the execution time is normalized to have directly

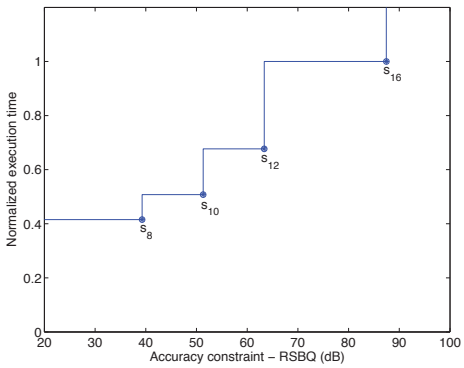

Fig. 1. Pareto curve for the SAD algorithm on 16×16 blocks.

the acceleration factor due to SWP operations.

For the other algorithms, the coordinates $(SQNR_x;C_x)$ associated with each point s_x of the Pareto curve are given in table 1. The Signal-to-Quantization Noise ratio is computed from the noise power P_b. The results show the ability of our approach to exploit the SWP capabilities associated with our architecture. Different trade-off between the implementation cost and the computation accuracy can be obtained. Thus, the reduction of the accuracy constraint offers opportunities to decrease the implementation costs. Compared to a classical processor having any SWP capabilities the maximal acceleration factor is equal to five when 8 bits data are used. As shown on the SATD example, the five supported data types can be completely exploited only when the number of processed data is important. For this example, only two distinct execution times are obtained because the size of the block lines is equal to four and thus only one packet of four data or two packets of two data can be used.

SWP configuration	s_8	s_{10}	s_{12}	s_{16}
SAD (16×16)	(39;54)	(51;66)	(63;88)	(87;130)
SAD (8×8)	(51;15)	(63;18)	(75;24)	(99;34)
SATD (4×4)	(20;40)	(32;40)	(44;80)	(68;80)
DCT 8	(34;256)	(46;256)	(58;384)	(82;512)
DCT 16	(37;2048)	(50;2048)	(62;3072)	(86;4096)

Table 1. Point s_x:$(SQNR_x;C_x)$ of the Pareto curve for different applications. The Signal-to-Noise Ratio $SQNR_x$ is expressed in dB and the cost C_x in cycles

To analyze the efficiency of our approach, the quality and the execution time of the global optimization process have been measured and compared with a classical *B&B* algorithm like the one presented in [8]. In this case, the search space is not limited to 3 values per variable and no initial value is available for the minimal cost. A first solution has to be found before applying the test on the minimal cost.

To analyze the quality, the solution obtained with the classical *B&B* algorithm is used as a reference. The relative error between the solution obtained with our approach and the reference solution has been measured for different accuracy constraints and different applications. For these different experiments, our global optimization approach always find the same solution as the classical *B&B* algorithm. The initial optimized solution and the refined solution are identical in 76% of the case. The maximal over-cost of the initial optimized solution is less than 9.6%.

The execution time has been measured for different numbers of variables in the optimization problem. The maximal execution times obtained for different accuracy constraints are reported in Table 2. The results for the initial optimized solution obtained with the heuristic and for the solution obtained after refinement are given. The execution time of the heuristic approach is very low and can be neglected compared to the refinement algorithm. Thus, the global execution time of our approach is closed to the one of the refinement algorithm. Our approach allows reducing significantly the execution time compared to the classical *B&B* algorithm. The heuristic algorithm allows finding a good optimized solution and can be used in stand-alone if the optimization execution time is a crucial parameter.

Maximal execution time ($sec.$)				
Number of variables	9	13	18	36
Initial optimized solution	0.3	0.45	1.09	2.4
Refined solution	0.4	6.3	38	44
Classical B& B	0.9	190	5212	5787

Table 2. Maximal optimization time (s) according to the number of variables to optimize

6. CONCLUSION

To implement complex multimedia applications in embedded systems, high performance, energy efficient and flexible architectures are needed. To reduce the time-to market, design automation tools are required to implement efficiently the applications on this kind of architectures. In this paper a new approach for word-length optimization in the case of architectures able to manipulate several data types has been presented. The search space of the *B&B* algorithm is drastically reduced thanks to an initial optimized solution obtained with a heuristic algorithm. Our approach allows efficiently benefit from the diversity of the data types supported by the architecture.

7. REFERENCES

[1] M.-A. Cantin, Y. Savaria, and P. Lavoie. A comparison of automatic word length optimization procedures. *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2:II–612–II–615 vol.2, 2002.

[2] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger. Accelerating Fixed-Point Design for MB-OFDM UWB Systems. *CommsDesign*, January 2005.

[3] M. Coors, H. Keding, O. Luthje, and H. Meyr. Fast Bit-True Simulation. In *IEEE/ACM Design Automation Conference 2001 (DAC 01)*, pages 708 – 713, Las Vegas, US, June 2001.

[4] M. Coors, H. Keding, O. Lthje, and H. Meyr. Design and DSP Implementation of Fixed-Point Systems. *EURASIP Journal on Applied Signal Processing*, 2002(9):908–925, 2002.

[5] S. Kim, K. Kum, and S. Wonyong. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. *IEEE Transactions on Circuits and Systems II*, 45(11):1455–1464, November 1998.

[6] K. Kum, J.Y. Kang, and W.Y. Sung. AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Transactions on Circuits and Systems II - Analog and Digital Signal Processing*, 47(9):840–848, September 2000.

[7] D. Menard, E. Casseau, S. Khan, O. Sentieys, S. Chevobbe, S. Guyetant, and R. David. Reconfigurable Operator Based Multimedia Embedded Processor. In *Reconfigurable Computing: Architectures, Tools and Applications*, volume 5453, pages 39–49, Karlsruhe Allemagne, 2009. Springer Berlin / Heidelberg.

[8] D. Menard, D. Chillet, and O. Sentieys. Floating-to-fixed-point Conversion for Digital Signal Processors. *EURASIP Journal on Applied Signal Processing, Special issue on Design Methods for DSP Systems*, 2006:1–19, january 2006.

[9] E. Raffin and al. Scheduling, binding and routing system for a run-time reconfigurable operator based multimedia architecture. In *Workshop on Design and Architectures for Signal and Image Processing DASIP 2010*, Edinburgh, 11 2010.

[10] R. Rocher, D. Menard, P. Scalart, and O. Sentieys. Analytical accuracy evaluation of Fixed-Point Systems. In *12th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, September 2007.