A Mixed Integer Linear Programming Approach for Design Space Exploration in FPGA-based MPSoC

Bouthaina Dammak, Rachid Benmansour and Smail Niar
LAMIH, University of Valenciennes
59300 Valenciennes, France

Mouna Baklouti and Mohamed Abid
National School of Engineers of Sfax
3042 Sfax, Tunisia

Abstract—Heterogeneous Multiprocessor System-on-Chip (Ht-MPSoC) architectures represent a promising approach as they allow a higher performance/energy consumption trade-off. In such systems, the processor instruction set is enhanced by application-specific custom instructions implemented on reconfigurable fabrics, namely FPGA. To increase area utilization and guarantee application constraint respect, we propose a new architecture where Ht-MPSoC hardware accelerators are shared among different processors in an intelligent manner. In this paper, a Mixed Integer Linear Programming (MILP) model is proposed to systematically explore the complex design space of the different configurations.

I. INTRODUCTION

The increase in resources in the latest FPGA generation, makes it possible to implement extremely complex Heterogeneous Multi-Processor System-on-Chip (Ht-MPSoC) architectures [1]. These architectures combine hardware and/or software cores, application specific hardware accelerators and communication units. Due to the increase in FPGA resources and the high number of architectural configurations, it becomes necessary to provide to the designer a Design Space Exploration (DSE) tool to determine the best architectural configuration for a given set of concurrent applications. In addition, this tool plays an important role in the design flow of an Ht-MPSoC architectures as it allows to determine the most efficient Ht-MPSoC configuration in a reduced time. This configuration is the one that requires the least FPGA resources in a reduced execution time and energy budget.

In the considered Ht-MPSoC, the system consists of multiple processors running software tasks and a group of HW accelerators that execute application specific instructions. These HW accelerators are shared between a given set of processors. The purpose of sharing HW accelerators between processors is to reduce circuit complexity in terms of logic elements while optimizing execution time and energy consumption.

In this paper, we consider Ht-MPSoC architectures with shared accelerators. In these architectures, the number of HW accelerators and their sharing type may vary from one processor to another. To explore the sharing space of these HW accelerators, we propose a Mixed Integer Linear Programming (MILP) formulation. Our model optimizes area usage of HW accelerators while respecting a performance constraint.

II. RELATED WORK

To tackle the problem of high area usage to integrate application-specific-instructions, most of the existing works propose heuristic approaches. In [2], Brisk et al. propose a polynomial-time heuristic that uses resource sharing to minimize the area required to synthesize a Set of custom Instruction Extension (ISEs). Their resource sharing approach transforms the set of ISEs into a single hardware data path.

Zuluaga et al. [3] introduce a latency constraints in the merging process of the ISEs to control the performance improvement they can make to a given application.

More recently, the work presented by Stojilović et al.[4] aims at a pragmatic increase in flexibility to integrate different ISEs from different applications. This works is motivated by the path-based datapath based algorithm presented in [2]. While [2] aims at minimizing the area cost, [4] increases the flexibility for a moderate cost. Their approach ensures that all ISEs from an application domain map on the same proposed domain-specific coarse-grained array.

The cited works provide an approach to share logic between different custom instructions for a single processor. Their proposed heuristic algorithms select the custom instructions to be mapped on logic providing a more area saving with operation sharing. Instead, we propose the sharing of an entire custom instruction between different cores and our MILP model explores the custom instructions to be mapped on hardware and the sharing-degree of hardware to support the ISE.

III. SHARING APPLICATION-SPECIFIC INSTRUCTIONS TECHNIQUE

Application-specific instructions are an effective way of improving the performance of processors. Critical computations can be accelerated based on new instructions executed on specialized hardware components.

An MPSoC on which N applications are running on the different processors has a less opportunity to implement all the application-specific instructions due to hardware resources constraint. Recent application tasks are based on same frequently used kernel functions such as matrix multiplication and convolution operations. For an Ht-MPSoC, without considering the hardware-sharing of similar kernels which are executed on different processors, custom accelerators will be implemented for different custom instructions that provide the
same computations. The proposed sharing approach offers a range of possible specific instructions shared among different tasks. This means that different computational tasks can share several accelerators. It is expected that this optimization will extenuate the area and power consumption and will preserve performance. We call a shared pattern the computational kernel operations existing on different applications. The shared pattern identification offers a range of possible optimization. In Figure 1, different applications have same heavy computational patterns (T1, T2 and T3 tasks) that can be executed simultaneously on private accelerators (one accelerator for each pattern/processor). However based on our proposed technique of sharing accelerators, only one accelerator for each pattern can be implemented and shared among the processors. The accelerators sharing and the sharing degree provide a large architectural space exploration.

IV. MIXED INTEGER LINEAR PROGRAMMING MODEL

Our space exploration addresses the way to merge the computational patterns, existing on the different applications, to reduce the overall area usage while respecting applications-performance constraints. Increasing the sharing degree reduces the area usage, but we may increase the delay of each processor to access shared accelerators and therefore we may not reach the required performance. Thus, the goal of our MILP model is to have minimum area usage, while keeping the execution time of each application under a required limit.

A. Problem Formulation

The architecture is a multi-processor system with $p$ processors running $p$ applications. These applications can be similar, i.e. Single Program Multiple Data model, or different, i.e. Multiple Program Single/Multiple Data. We define a pattern, a time consuming task-kernel existing in one or different tasks. Based on expected acceleration, we select the sequence of most computational patterns executed on the $p$ processors. These patterns are candidates for hardware implementations that may fulfill the desired performance. Let $\{T_1, \ldots, T_m\}$ denotes this sequence and $P = \{P_1, P_2, \ldots, P_i, \ldots, P_n\}$ the sequence of $n$ homogeneous processors. Let $N = \{1, 2, \ldots, n\}$ and $M = \{1, 2, \ldots, m\}$.

Each consuming pattern $T_j$ ($j \in M$), is assigned a predefined area constant $a_j$ the value of which is the number of FPGA area required by $T_j$ to be implemented as hardware accelerator. In addition, we define two integer constants $t_{s,j,i}$ and $t_{e,j,i}$ respectively for the start-time and the end-time of the pattern $T_j$ on processor $P_i$.

The implementation of $T_j$ in private way provides a defined acceleration $t_{acc,j}$. For each processor $P_i$, an acceleration constant $limit_i$ is set as a constraint for the total processor acceleration $acc_i$.

B. Objective Function

In this section we present a MILP formulation of the problem so that we can obtain an optimal solution with the help of a commercial optimization solver for mixed integer linear programming. Let $x_{ij}$ a binary variable that denotes whether $T_j$ is implemented on Hardware (HW) for processor $P_i$:

$$x_{ij} = \begin{cases} 1, & \text{if } T_j \text{ is implemented on HW} \\ 0, & \text{else} \end{cases}$$

Let $y_{ijk}$ a binary variable that denotes whether the accelerator (Acc) of task $T_j$ is shared between processors $P_i$ and $P_k$ or not.

$$y_{ijk} = \begin{cases} 1, & \text{if Acc of } T_j \text{ is shared between } P_i \text{ and } P_k \\ 0, & \text{otherwise} \end{cases}$$

Our objective function is to minimize the total area required to implement the $m$ patterns.

$$Total\_Area = \sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} \sum_{k=1}^{n} a_j y_{ijk}$$

(1)

To linearise equation 1, we define new continuous variables $z_{ij}$ and $w_{ij}$

$$z_{ij} = \frac{1}{\sum_{k=1}^{n} y_{ijk}} = \frac{1}{sh_{ij}}$$

$$w_{ij} = z_{ij} x_{ij}$$

The definition of $z_{ij}$ can be expressed in linear form as follows:

$$z_{ij} + \sum_{k=1}^{n} y_{ijk} = 1$$
\[ z_{ij} + \sum_{k=1}^{n} \theta_{ijk} = 1 \]  

(2)

Where \( \theta_{ijk} \) is a continuous variable expressed as follows \( \theta_{ijk} = z_{ij}y_{jik} \) and satisfying the following constraints:

\[
\begin{align*}
\theta_{ijk} &\leq y_{jik} \\
\theta_{ijk} &\leq z_{ij} \\
\theta_{ijk} &\geq z_{ij} + y_{jik} - 1
\end{align*}
\]

(3)

The objective function (eq. (1)) can be re-written as:

\[ \text{MinAir} = \sum_{j=1}^{m} \sum_{i=1}^{n} a_j w_{ij} \]  

(4)

C. Performance constraint

The performance constraint is based on the following principle: the total acceleration for each processor provided by the mapping of the different tasks on hardware accelerators needs to be upper a performance limit. Regarding the sharing degree of hardware accelerator, each processor has a delay \( R_{ji} \) to access this shared accelerator.

The performance constraint can be imposed as follows:

\[ acc_i = \sum_{j=1}^{m} x_j acc_j - x_j R_{ji} \geq limit_i, \forall i \in N \]  

(5)

\[ R_{ji} = t_{e,jk} - t_s^{h} \]  

where \( k = \text{max}\{0, 1, ..., i - 1\} \) and \( y_{jik} = 1 \)

\[ R_{ji} = \sum_{k=1}^{i-1} p_{jik}(t_{e,jk} - t_{s,jk}) \]  

(6)

Where \( t_{e,jk} \) and \( t_{s,jk} \), \( j \in M, i \in N, k \in \{1, 2, \ldots, i\} \), are continuous variables that define respectively the start-time and the end-time of executing \( T_j \) on hardware (HW) respectively on processors \( P_k \) and \( P_i \) and are calculated as follow:

\[ \forall j \in M, \forall i \in N, \forall k \in \{1, 2, \ldots, i\}, \]

\[ t_{s,jk} = t_{s,j} - \sum_{l=1}^{j-1} x_l (acc_l - R_{li}) \]

\[ t_{e,jk} = t_{e,j} - \sum_{l=1}^{j} x_l (acc_l - R_{lk}) \]

\( p_{jik}, j \in M, (i, k) \in N^2 \), is a binary variable defined as follow:

\[ p_{jik} = \begin{cases} 
1, & \text{if } P_k \text{ is the last processor sharing } T_j \text{ with } P_i \\
0, & \text{else}
\end{cases} \]

To add the definition of \( p_{jik} \) variable to our MILP model, the following assumptions are incorporated as constraints:

1) If processors \( P_k \) and \( P_i \) haven’t a shared hardware accelerator of task \( T_j \) \( (y_{jik} = 0) \) then \( p_{jik} \) will be equal to zero.

\[ p_{jik} \leq y_{jik} \forall j \in M, \forall (i, k) \in N^2 \]

2) For a processor \( P_i \) and a task \( T_j \), the sum of \( p_{jik} \) over \( k \) is equal to 1 or 0

\[ \sum_{k=1}^{n} p_{jik} \leq 1 \forall j \in M, i \in N \]

3) If \( P_k \) is the last processor sharing with \( P_i \) the hardware implementation of task \( T_j \) then \( p_{jik} \) will be equal to 1.

In other words

\[ \forall j \in M, \forall i \in N, \forall k \in \{1, 2, \ldots, i\} \]

if \( (y_{jik} - \sum_{l=k+1}^{i} y_{jil} \geq 1) \) Then \( p_{jik} \geq 1 \)

In a linear form, this assumption can be expressed as follows:

\[ \forall j \in M, \forall i \in N, \forall k \in \{1, 2, \ldots, i\} \]

\[ y_{jik} - \sum_{l=k+1}^{i} y_{jil} + V1 \cdot r_{jik} \geq 1; \]

\[ p_{j,i,k} + V2 \cdot r_{j,i,k} >= 1; \]

\[ V3(1 - r_{j,i,k}) \geq y_{j,i,k} - \sum_{l=k+1}^{i} y_{jil}; \]

Where \( V1, V2 \) and \( V3 \) are large constants and \( r_{j,k} \) is a binary variable.

4) If \( P_k \) is not the last processor sharing with \( P_i \) the hardware implementation of task \( T_j \) then \( p_{jik} \) will be equal to 0:

This assumption is expressed as an IF-THEN constraint:

\[ \forall j \in M, \forall i \in N, \forall k \in \{1, 2, \ldots, i\} \]

If \( \sum_{l=k+1}^{i} y_{jil} \geq 1 \) Then \( p_{jik} \leq 0 \)

In linear form, this constraint is expressed as follows

\[ \forall j \in M, \forall i \in N, \forall k \in \{1, 2, \ldots, i\} \]

\[ \sum_{l=k+1}^{i} y_{jil} + V1 \cdot q_{jik} \geq 1; \]

\[ 1 - p_{j,i,k} + V2 \cdot q_{j,i,k} \geq 1; \]

\[ V3(1 - q_{j,i,k}) \geq \sum_{l=k+1}^{i} y_{jil}; \]
Area usage of the generated solutions for different timing constraints

Fig. 2. Area usage of the MILP generated configuration for different speed-up

Where \( q_{jik} \) is a binary variable.

Now the period constraint can be re-written as:

\[
\forall i \in N \quad \text{acc}_i = \sum_{j=1}^{m} t_{\text{acc}j} - \sum_{k=1}^{i-1} p_{jik} \times (te_{hw_{jk}} - ts_{hw_{ji}}) \geq \text{limit}_i \tag{7}
\]

V. EXPERIMENTAL RESULTS

For our experiments, we focus on generating the best configuration for a 4 and 8 multiprocessor architecture executing different applications. Each processor applications consists of three computational patterns \( \{T_1, T_2, T_3\} \) and a number of non-computational loops to obtain different applications.

Table I summarizes the execution time and the area requirement for the three tasks. Note that the area requirement is presented in terms of area unit. An area unit corresponds to 150 slices. Figure 2 shows the area usage variation of the MILP generated configuration when varying the speed-up. We assume that all processors require the same speed up. As expected, the area usage of the resulting configuration increases when we increase the speed-up. In addition, we notice the same area usage for different configurations providing different speed-ups. For example, in figure 2.a, the configuration that provides a speed-up equal to 1.75 requires the same area as the configuration that provides 2.06. These configurations require two HW accelerators of task \( T_3 \). For the first configuration, \( P_1 \) and \( P_2 \) share a HW accelerator and \( P_3 \) and \( P_4 \) share the second accelerator. For the second configuration, processors \( P_1 \) and \( P_3 \) share a HW accelerator and \( P_2 \) and \( P_4 \) share the second accelerator. We deduce that different combination of the processors sharing a task could vary the performance. For the 4 and 8 processors architectures we note that the maximum speed-up is provided with a reduced area-usage configuration than the private configuration. The 4-processor architecture with private \( T_1, T_2 \) and \( T_3 \) accelerators provides a speed-up equal to 2.9 and consumes 84 area units.

As shown in Figure 2, for the maximum speed-up, our model generates a configuration that consumes 63 area units.

TABLE I

<table>
<thead>
<tr>
<th>T1: Data inversion loop</th>
<th>Area usage</th>
<th>SW/HW execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2: Loop multiplication</td>
<td>15</td>
<td>3815,213</td>
</tr>
<tr>
<td>T3: Find maximum</td>
<td>4</td>
<td>2000,1200</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, we propose an efficient MILP model to optimize the area usage of FPGA-based Hi-MPSOC shared application-specific instructions. The proposed MIP model leads to the optimal application-specific instructions sharing degree that optimize the architecture resource allocation and satisfy the performance constraint. Experimental results confirm the efficiency of our approach. For the maximum speed up, the MIP model generates a reduced area usage configuration compared to the private one.

ACKNOWLEDGMENT

The authors would like to thank International Campus on Safety and Intermodality in transportat (CISIT in French) for the support given to this project.

REFERENCES