Abstract

The PinHaT software allows the automatic generation of FPGA-based multiprocessor systems from parallel programs using Integer Linear Programming for high-level synthesis and the vendor's tool-chain.

1. Introduction

As apparent in current developments the reduction of transistor size and the exploitation of instruction-level parallelization can no longer be continued to enhance the performance of processors. Instead, multi-processor systems are a common way of enhancing performance by exploiting parallelism of applications. This project deals with the automated design of multiprocessor systems from parallel applications using combinatorial optimization. The underlying problem is simultaneous task-mapping and architecture determination.

In section 2 the architectural synthesis flow is presented and synthesis results are presented in section 3. An overview about the PinHaT software is given in section 4 and finally this paper is concluded in section 5.

2. The Synthesis Flow

In figure 1 the underlying design methodology of PinHaT is shown. The flow starts with a parallel application, from which the application-specific architecture is synthesized [1].

The parallel application, using the message passing paradigm (MPI), is simulated and analyzed to obtain information about task precedence and inter-task data traffic. This information is used to specify an instance of an Integer Linear Program (ILP) problem. Additionally, constraints of processing elements and communication networks, for a target platform are needed. That is, in the case of Xilinx FPGAs for instance the actual size of the FPGA, costs for the processing elements, like MicroBlaze or PowerPC processor or the bandwidth and latency of a communication networks. The instance of the ILP problem is then solved using the lp_solve 5.5 library. The result is a concrete system description, including both software and hardware. That is, all tasks are mapped to processors and all processors are connected via networks. The flow ends with the appropriation of the concrete system information which is processed by the vendor’s tool chain, e.g. Xilinx EDK or Altera Quartus II. The results of logic synthesis and placement and routing can be used to extract cost models and platform constraints.

Figure 1: Architectural Synthesis Flow

A lightweight and flexible MPI library, called SoC-MPI [3], is used to take care of the software-sided on-chip communication. SoC-MPI currently supports a subset of the MPI standard and allows hardware accelerated MPI functions. The library is divided into two layers, a network-dependent layer and a network-independent layer. This separation hides enough complexity from the user, without loosing too much performance due to high degree of abstraction.

3. Synthesis Results

To demonstrate the feasibility of the proposed method, several experiments were conducted with a parallel implementation of a signal processing chain for the IEEE 802.11g WLAN standard. The Message Passing Interface (MPI) standard was used in the implementation. However, only a subset of the standard that can be efficiently implemented in embedded systems has been used. The implementation performs, in the order, the following: timing synchronization, coarse and fine frequency offset estimation and compensation, symbol demapping, FFT (OFDM demodulation), pilot extraction, channel estimation and compensation, carrier phase offset mapping, and finally timing drift estimation and compensation [4].
The resulting architecture is shown in figure 3. The goal was to minimize the last finishing time of the tasks (the makespan). Therefore the Critical Path (CP) of the task graph must be minimized. However, in this problem, the CP is not always distinct, as it depends on the heterogeneous architecture which is not known a priori. Moreover, it is necessary to consider preemptive scheduling on a processor for real-time systems when several tasks are mapped on one processor. Without real-time scheduling on processors, all tasks were mapped on one core of type P3. With real-time scheduling, three cores were used (see figure 3). Each task can communicate via more than one medium, so that two networks connect any pair of processors.

Table 1: Task functionality and deadlines

<table>
<thead>
<tr>
<th>Function</th>
<th>Index</th>
<th>Deadline (ns)</th>
<th>P1 (ns)</th>
<th>P2 (ns)</th>
<th>P3 (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>10</td>
<td>19</td>
<td>6032</td>
<td>1034</td>
<td>11</td>
</tr>
<tr>
<td>phase</td>
<td>11</td>
<td>988</td>
<td>3504</td>
<td>601</td>
<td>601</td>
</tr>
<tr>
<td>channel</td>
<td>12</td>
<td>988</td>
<td>9099</td>
<td>156</td>
<td>156</td>
</tr>
<tr>
<td>timing</td>
<td>13</td>
<td>19</td>
<td>33781</td>
<td>5791</td>
<td>17</td>
</tr>
<tr>
<td>demap</td>
<td>14</td>
<td>1216</td>
<td>598</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>FFT</td>
<td>15</td>
<td>1216</td>
<td>534</td>
<td>334</td>
<td>534</td>
</tr>
<tr>
<td>fine</td>
<td>16</td>
<td>19</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>course</td>
<td>17</td>
<td>19</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

The functionality of each task is given in table 1. The table also shows the deadlines of the tasks as derived from the standard. In this implementation, the deadlines are equal to the periods. The three last columns in the table show approximately execution time in nanoseconds of the tasks on three different processors with loosely coupled accelerators.

The presented case study with a parallel implementation of a signal processing chain for IEEE 802.11g standard demonstrates the feasibility of the method. Several experiments were conducted and lead to good solutions. However, it is not possible to show all results in this paper. Detailed timings can be read in [4].

4. The PinHaT software

The PinHaT software is written in JAVA and integrates the described synthesis flow in an easy to use environment (see figure 2) [2]. Users have two possibility to generate the system (hardware and software). Firstly, by synthesizing the architecture as described in section 2 and secondly by defining and subsequently refining an abstract system specification. Therefore XML, in conjunction with document type definition, is used to represent the system. A mapper creates the platform dependent hardware description files, which is passed to the vendor's tool-chain for logic synthesis and placement and routing.

5. Conclusion

The PinHaT software eases the design of multiprocessor systems-on-chip using FPGAs by allowing the automated architecture synthesis from a parallel application. High-level synthesis and the vendor tool-chain are used to generate the FPGA configuration-file.

6. References


