Decomposition based Methods for Allocation and Scheduling Problems arising in System Design

Luca Benini
*Michele Lombardi
Michela Milano
Martino Ruggiero

DEIS, University of Bologna
Talk topic & outline

Talk topic
• Context: CAD tools for applications running on MPSoCs...
• ...solution techniques for a core optimization problem (resource allocation & scheduling)
• Practical example: a CAD tool for the Cell BE platform

Outline
• Context & Problem description
• Solution ingredients
• Solution methods
  • Multi Stage Logic based Benders’ Decomposition
  • Constraint Programming
  • A hybrid solver
1. Context & problem
Embedded System:
an information processing device embedded into another product

- Often dedicated to a specific application
- Often operate real time tasks
- Parallelism to contain energy consumption

Multicore platforms
Design flow

It’s tricky to exploit parallelism! ➔ Need for design tools

Task Graph

```
... int s = 0; for(i=0; i<n; i++){ s = pow(s, i); }
...```

• Task to µP mapping
• Resource allocation
• Scheduling

arc = data comm.
Design flow

It’s tricky to exploit parallelism! Need for design tools

```
... int s = 0; for(i=0; i<n; i++){ s = pow(s, i); ...

task = process
```

Task Graph

```
... arc = data comm.
```

Optimization component

- Task to μP mapping
- Resource allocation
- Scheduling
Cell Broadband Engine: High performance multicore μP

- General purpose Power Processing Element (PPE)
- On Chip memory (DRAM)
- 8 Synergistic Processing Elements (SPE)
  - GP Processors
  - Primed by PPE
  - Small memory

Exploiting the full power of Cell is very tricky
Problem specification

A task graph

The CELL platform

Time behavior
Problem specification

A task graph

The CELL platform

OUTPUT: spe & mem allocation + schedule

MAKESPAN
A task graph

@TASK_GRAPH 0 {
  PERIOD 1000000
  TASK t0_0 TYPE 0
  TASK t0_1 TYPE 1
  TASK t0_2 TYPE 2
  ARC a0_0 FROM t0_0 TO t0_1 TYPE 0
  ARC a0_1 FROM t0_0 TO t0_2 TYPE 1
}

@TRANS 0 {
  #type comm rd_ls rd_lr rd_rr wr_ls wr_lr wr_rr
  0 8033 171 212 248 113 152 190
  1 3468 135 163 218 105 153 195
}

@PE 0 {
  #type version dur ext_dur comp_mem
  0 0 1708 1827 1213
  1 0 1837 1947 2143
  2 0 1240 1351 4807
}

The CELL platform

@PLATFORM {
  # clock bus_bandwidth
  1 1000
  #id capacity
  0 100000
  1 100000
  2 100000
  3 100000
  4 100000
  5 100000
}
2. Solution “ingredients”
Solution ingredients: MILP

- It's a declarative programming paradigm
- Problem =

\[
\begin{align*}
\min \ z &= c^T x \\
\text{s.t.} \ Ax &\geq b \\
x_i &\geq 0 \\
x_i \text{ integer} \quad \forall i \in I
\end{align*}
\]

Efficient algorithms are available
- Simplex, interior point
- Branch & bound, branch & cut
- ...
Introduction to Constraint Programming

- It’s another declarative programming paradigm
- Mainly targets Constraint Satisfaction Problems (NP-hard)

\[ \text{CSP} = \langle X, \ D, \ C \rangle \]
- \(X\) is a set of variables
- \(D\) is the set of their domains
- \(C\) is a set of constraints (any!)

All constraints embed a filtering algorithm to remove inconsistencies

A sample problem: scheduling a TG

- No communications
- Each task has a fixed duration \(\text{dur}_i\)
- Each task uses a processor
- Deadline \(d_l\)
Modeling with CP

CSP = <X, D, C>

For each task: \( \text{dur}_i \)
Deadline: \( \text{dl} \)

Variables & domains:

\( \text{START}_i \in \{0, \ldots, \text{dl}\} \)

\( \text{END}_i \in \{0, \ldots, \text{dl}\} \)

Constraints:

\( \forall i \text{ END}_i = \text{START}_i + \text{dur}_i \)

\( \text{END}_0 \leq \text{START}_1, \text{ END}_1 \leq \text{START}_3 \)

\( \text{END}_2 \leq \text{START}_3 \)

\( \text{cumulative}([\text{START}_0, \text{START}_2], [\text{dur}_0, \text{dur}_2], [1,1], 1) \)

\( \text{cumulative}([\text{START}_1, \text{START}_3], [\text{dur}_1, \text{dur}_3], [1,1], 1) \)
Solving a CP problem

**Depth First Search**

- START$_2$=0
- POST $t_2$
- START$_0$=0
- WAKE UP $t_2$
- START$_1$=1
- START$_3$=2
- OK!

**Constraints:**

\[
\forall i \quad \text{END}_i = \text{START}_i + \text{dur}_i \\
\text{END}_0 \leq \text{START}_1, \quad \text{END}_1 \leq \text{START}_3 \\
\text{END}_2 \leq \text{START}_3 \\
\text{cumulative}([\text{START}_0, \text{START}_2])
\]

**Data:**

dur$_0$=1, dur$_1$=1, dur$_2$=2, dur$_3$ = 1, dl = 4

**Domains:**

\[
\begin{align*}
\text{START}_0 &\in \{0,1,2,3,4\} \\
\text{END}_0 &\in \{0,1,2,3,4\} \\
\text{START}_1 &\in \{0,1,2,3,4\} \\
\text{END}_1 &\in \{0,1,2,3,4\} \\
\text{START}_2 &\in \{0,1,2,3,4\} \\
\text{END}_2 &\in \{0,1,2,3,4\} \\
\text{START}_3 &\in \{0,1,2,3,4\} \\
\text{END}_3 &\in \{0,1,2,3,4\}
\end{align*}
\]

**Optimality via a sequence of feasibility problems**
3. Solution methods
Problem features
- Large allocation decision space
- Makespan objective

Makespan objective

Good back-propagation

Tackle with OR techniques
Tackle with CP

Use LBD
- Hooker (’95)
- Allows to mix solvers
- SubP relaxations
- CUTS

#1: Logic Based Benders’ Decomposition

\[ \approx n^8 \times 2^n \times 2^2 \times 3^2 \]
#1: Logic Based Benders’ Decomposition

**Problem features**

- Large allocation decision space
- Makespan objective

**Integer linear model**

\[
\begin{align*}
T_{ij} &= 1 \text{ if } t_i \text{ on } \text{SPE}_j \\
M_{ij} &= 1 \text{ if comp. data on } \text{SPE}_j \\
R_{rj} &= 1 \text{ buffer local to reader} \\
W_{rj} &= 1 \text{ buffer local to writer}
\end{align*}
\]

**CP model**

START, END vars
Cumulative constraints

**LBD**

- SPE & mem Allocation
- Scheduling

end

feas? yes

store sol

cut no
• **PRO:** - much simpler and easier subproblems

• **CON:** - hard to design good high level objective functions
  - loosely couple subproblems
Cuts for complex problems

Most basic cut: **nogood**

\[ \sigma: \begin{array}{cccc}
T_{00} = 1 & T_{01} = 0 & T_{10} = 1 & \ldots & M_{00} = 1 & M_{01} = 0 & \ldots & R_{00} = 1 & \ldots & W_{00} = 1
\end{array} \]

\[ \sum_{\sigma(T_{ij}) = 1} (1 - T_{ij}) + \sum_{\sigma(M_{ij}) = 1} (1 - M_{ij}) + \sum_{\sigma(M_{ij}) = 0} M_{ij} + \ldots > 0 \]

**TOO WEAK**  
**strengthen via a refinement procedure**

- Select a subset of decisions (set of vars)
- Relax consequences & solve the problem again
- If still infeasible => remove vars from cut

\[ T_M = \alpha \cdot T_S \Rightarrow \text{SP can be safely solved } \alpha \text{ times} \]

- Iterative methods: de Siqueira and Puget, Junker
- **Requires to solve** \( O(n^*\log(n)) \) relaxed NP-hard problems
As for the CONS?

- Best solutions are found late
- Bad makespan estimation at higher levels

Pure CP approach

- No decomposition
- Exploit propagation to compute makespan bounds

In particular

- Quite straightforward model
- Carefully designed search strategy

#2: pure CP approach
CP Model

• SPE allocation:

\[ T_i = j \text{ iff } t_i \text{ on } \text{SPE}_j \]

• MEM allocation:

\[ M_i = 1 \text{ iff } t_i \text{ locally allocates comp. Data} \]
\[ R_r = 1 \text{ iff buffer local to reader} \]
\[ W_r = 1 \text{ iff buffer local to writer} \]

buffer allocation constraints
memory capacity constraints

• Scheduling:

START, END vars for each operation (rd, ex, wr)
cumulative constraints for SPEs
CP Search strategy

All task ordered?

Choose task

Sort by sched. time

To SPE\(_0\)^* \\
To SPE\(_1\)^* \\
To SPE\(_2\)^* \\

To SPE\(_0\)^*

dur = dur\(_0\)

dur = dur\(_1\)

queue

postpone

SPE\(_0\)

SPE\(_1\)

1

2

3

4

T1

T2

W2-3

W2-4

R2-3

T3

R2-4

T4
Thrashing...

An early bad decision sunks the performance

- randomization
- Frequent restarts

CP cuts

- During CP search cuts are generated and refined
- A basic cut is an infeasible SPE & MEM allocation
Experimental setup

Three groups of instances

1. Real (90) synth benchmark – memory impact negligible
2. Real-like, communication intensive (100)
3. Real-like, computation intensive (100)

Graph generator
- Random to realistic struc.
- Attribute dependencies
- TGFF file format

One platform
1. CELL BE
2. 6 SPE available

Negligible communication durations

very poor impact of memory allocation...

Same graph structure, artificial durations and requirements
Results for group 1

Memory allocation has no impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>&gt; TL</th>
<th>time</th>
<th>&gt; TL</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>9-14</td>
<td>0</td>
<td>0.42</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>15</td>
<td>14-27</td>
<td>0</td>
<td>0.57</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>25</td>
<td>30-56</td>
<td>1</td>
<td>80.88</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>25</td>
<td>56-66</td>
<td>2</td>
<td>274.39</td>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>30</td>
<td>47-72</td>
<td>5</td>
<td>354.81</td>
<td>1</td>
<td>0.90</td>
</tr>
<tr>
<td>30</td>
<td>73-83</td>
<td>7</td>
<td>280.02</td>
<td>0</td>
<td>0.41</td>
</tr>
</tbody>
</table>
Results for group 2

Memory allocation has strong impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>&gt; TL</th>
<th>time</th>
<th>&gt; TL</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-11</td>
<td>6-14</td>
<td>0</td>
<td>4.01</td>
<td>4</td>
<td>4.68</td>
</tr>
<tr>
<td>12-13</td>
<td>8-15</td>
<td>0</td>
<td>6.32</td>
<td>5</td>
<td>0.06</td>
</tr>
<tr>
<td>14-15</td>
<td>12-19</td>
<td>0</td>
<td>5.54</td>
<td>5</td>
<td>180.16</td>
</tr>
<tr>
<td>16-17</td>
<td>15-22</td>
<td>0</td>
<td>28.35</td>
<td>6</td>
<td>226.66</td>
</tr>
<tr>
<td>18-19</td>
<td>17-24</td>
<td>0</td>
<td>105.50</td>
<td>7</td>
<td>10.74</td>
</tr>
<tr>
<td>20-21</td>
<td>21-29</td>
<td>1</td>
<td>210.89</td>
<td>10</td>
<td>---</td>
</tr>
<tr>
<td>22-23</td>
<td>21-30</td>
<td>2</td>
<td>388.00</td>
<td>9</td>
<td>250.00</td>
</tr>
<tr>
<td>24-25</td>
<td>24-35</td>
<td>3</td>
<td>268.57</td>
<td>9</td>
<td>85.00</td>
</tr>
<tr>
<td>26-27</td>
<td>27-39</td>
<td>4</td>
<td>375.00</td>
<td>8</td>
<td>160.49</td>
</tr>
<tr>
<td>28-29</td>
<td>32-43</td>
<td>5</td>
<td>528.00</td>
<td>6</td>
<td>432.25</td>
</tr>
</tbody>
</table>
Results for group 3

Buffer allocation has no impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>MS-LBD</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&gt; TL</td>
<td>&gt; TL</td>
</tr>
<tr>
<td>10-11</td>
<td>6-14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12-13</td>
<td>8-15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14-15</td>
<td>12-19</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16-17</td>
<td>15-22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18-19</td>
<td>17-24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20-21</td>
<td>21-29</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22-23</td>
<td>21-30</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24-25</td>
<td>24-35</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>26-27</td>
<td>27-39</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>28-29</td>
<td>32-43</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>
Some considerations

• MS-LBD solver is robust due to the use of strong cuts...
• ...but it cannot effectively exploit makespan bounds
• CP solver can exploit and compute makespan bounds...
• ...it seems more capable to quickly find good solutions...
• ...but it cannot deal with buffer allocation

We could build another hybrid solver!
Structure of the new hybrid (LBD like)

- Both solvers can improve solution & prove optimality
- Cuts greatly help the CP solver
Results for group 1

Memory allocation has no impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>MS-LBD</th>
<th>CP</th>
<th>HYB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&gt; TL</td>
<td>time</td>
<td>&gt; TL</td>
</tr>
<tr>
<td>15</td>
<td>9-14</td>
<td>0</td>
<td>0.42</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>14-27</td>
<td>0</td>
<td>0.57</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>30-56</td>
<td>1</td>
<td>80.88</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>56-66</td>
<td>2</td>
<td>274.39</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>47-72</td>
<td>5</td>
<td>354.81</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>73-83</td>
<td>7</td>
<td>280.02</td>
<td>0</td>
</tr>
</tbody>
</table>
Results for group 2

Memory allocation has strong impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>MS-LBD &gt; TL</th>
<th>MS-LBD time</th>
<th>CP &gt; TL</th>
<th>CP time</th>
<th>HYB &gt; TL</th>
<th>HYB time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-11</td>
<td>6-14</td>
<td>0</td>
<td>4.01</td>
<td>4</td>
<td>4.68</td>
<td>0</td>
<td>8.03</td>
</tr>
<tr>
<td>12-13</td>
<td>8-15</td>
<td>0</td>
<td>6.32</td>
<td>5</td>
<td>0.06</td>
<td>0</td>
<td>6.38</td>
</tr>
<tr>
<td>14-15</td>
<td>12-19</td>
<td>0</td>
<td>5.54</td>
<td>5</td>
<td>180.16</td>
<td>0</td>
<td>6.79</td>
</tr>
<tr>
<td>16-17</td>
<td>15-22</td>
<td>0</td>
<td>28.35</td>
<td>6</td>
<td>226.66</td>
<td>0</td>
<td>9.22</td>
</tr>
<tr>
<td>18-19</td>
<td>17-24</td>
<td>0</td>
<td>105.50</td>
<td>7</td>
<td>10.74</td>
<td>0</td>
<td>54.12</td>
</tr>
<tr>
<td>20-21</td>
<td>21-29</td>
<td>1</td>
<td>210.89</td>
<td>10</td>
<td>---</td>
<td>3</td>
<td>52.08</td>
</tr>
<tr>
<td>22-23</td>
<td>21-30</td>
<td>2</td>
<td>388.00</td>
<td>9</td>
<td>250.00</td>
<td>5</td>
<td>63.79</td>
</tr>
<tr>
<td>24-25</td>
<td>24-35</td>
<td>3</td>
<td>268.57</td>
<td>9</td>
<td>85.00</td>
<td>3</td>
<td>42.71</td>
</tr>
<tr>
<td>26-27</td>
<td>27-39</td>
<td>4</td>
<td>375.00</td>
<td>8</td>
<td>160.49</td>
<td>3</td>
<td>168.64</td>
</tr>
<tr>
<td>28-29</td>
<td>32-43</td>
<td>5</td>
<td>528.00</td>
<td>6</td>
<td>432.25</td>
<td>5</td>
<td>56.98</td>
</tr>
</tbody>
</table>
Results for group 3

Buffer allocation has no impact on durations

<table>
<thead>
<tr>
<th>tasks</th>
<th>arcs</th>
<th>MS-LBD</th>
<th>CP</th>
<th>HYB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&gt; TL</td>
<td>time</td>
<td>&gt; TL</td>
</tr>
<tr>
<td>10-11</td>
<td>6-14</td>
<td>0</td>
<td>0.21</td>
<td>0</td>
</tr>
<tr>
<td>12-13</td>
<td>8-15</td>
<td>0</td>
<td>1.16</td>
<td>0</td>
</tr>
<tr>
<td>14-15</td>
<td>12-19</td>
<td>0</td>
<td>1.00</td>
<td>0</td>
</tr>
<tr>
<td>16-17</td>
<td>15-22</td>
<td>0</td>
<td>10.89</td>
<td>0</td>
</tr>
<tr>
<td>18-19</td>
<td>17-24</td>
<td>0</td>
<td>48.92</td>
<td>0</td>
</tr>
<tr>
<td>20-21</td>
<td>21-29</td>
<td>1</td>
<td>116.10</td>
<td>1</td>
</tr>
<tr>
<td>22-23</td>
<td>21-30</td>
<td>1</td>
<td>69.16</td>
<td>0</td>
</tr>
<tr>
<td>24-25</td>
<td>24-35</td>
<td>3</td>
<td>269.57</td>
<td>0</td>
</tr>
<tr>
<td>26-27</td>
<td>27-39</td>
<td>7</td>
<td>88.67</td>
<td>3</td>
</tr>
<tr>
<td>28-29</td>
<td>32-43</td>
<td>8</td>
<td>310.00</td>
<td>4</td>
</tr>
</tbody>
</table>
Conclusion & future works

In conclusion...

1. A number of approaches to solve an important allocation & scheduling problem arising in CAD tools for MPSoCs
2. Nice CP features: integration & side constraints easily added
3. Hybridization & decomposition can be used to combine the strengths and minimize the weaknesses of different solvers

Some possible developments...

1. The CELL flow model has changed: adapt the best approaches
2. Yet improve the solver (in particular the CP one)
3. Different instances are best tackled with different solvers => machine learning
4. ...any suggestion?
Thank you!

Questions?
Decomposition based Methods for Allocation and Scheduling Problems arising in System Design

Luca Benini
*Michele Lombardi
Michela Milano
Martino Ruggiero

DEIS, University of Bologna