A Multiobjective Reconfiguration-Aware Scheduler for FPGA-Based Heterogeneous Architectures: extended MILP formulation

Enrico A. Deiana, Marco Rabozzi, Riccardo Cattaneo, Marco D. Santambrogio
Politecnico di Milano, Milan, Italy
enrico.deiana@mail.polimi.it, {marco.rabozzi, riccardo.cattaneo, marco.santambrogio}@polimi.it

This technical document presents an extended description of the Mixed-Integer Linear Programming (MILP) formulation used for solving the scheduling problem on heterogeneous architectures consisting of software components and reconfigurable logic. The following sections describe the variables, parameters, constraints, objective function and cutting planes for the formulation.

A. Sets and parameters

In order to define the MILP model we need to introduce several sets related to the problem:

$T :=$ set of tasks to schedule;
$RT :=$ set of reconfiguration tasks;
$AT :=$ set of all tasks;
$I^s :=$ set of software implementations;
$I^h :=$ set of hardware implementations;
$I :=$ set of all the available implementations;
$C^s :=$ set of software components;
$C^h :=$ set of hardware components;
$C :=$ set of all the available components;
$P :=$ set of tasks dependencies (i.e. $(t1, t2) \in P$ if and only if $t2$ depends directly on $t1$);
$P^+ :=$ the transitive closure of P (i.e. $(t1, t2) \in P^+$ if and only if $t2$ depends on $t1$);
$R :=$ set of Field Programmable Gate Array (FPGA) resources (i.e. CLB, DSP, BRAM...).

Notice that we do not know beforehand which is the number of reconfiguration tasks that will be performed in the final schedule. However, the set RT must have a fixed number of elements that cannot vary during the optimization of the MILP model. In order to solve this issue, we consider the worst case scenario in which each task, except for the first one, requires a reconfiguration before its execution, so that overall $|T| - 1$ reconfigurations are performed. Within the set RT we consider $|T| - 1$ elements, while a special binary variable ($rtt_{rt,t}$) for each element will be used to determine if the element represents a required reconfiguration or not. The following are the parameters needed for the MILP model:

$T_{max} :=$ maximum time for the execution of the schedule;
$T_\epsilon :=$ minimum time unit;
$time_i :=$ execution time of implementation $i \in I$;
$power_i :=$ power consumption of implementation $i \in I$;
$energy_i :=$ energy consumption of implementation $i \in I$;
$map_{t,i,c} :=$ 1 if task $t \in T$ can be mapped on component $c \in C$ with implementation $i \in I$, 0 otherwise;
$res_{i,r} :=$ resources of type $r \in R$ required by implementation $i \in I^h$;
$maxRes_r :=$ number of resources of type $r \in R$ within the FPGA;
$bit_r :=$ average bitstream size for a resource of type r;
$bit_{max} :=$ maximum bitstream size for reconfiguration;
$T_{rec} :=$ reconfiguration time for each unit of bitstream;
$P_{rec} :=$ power consumption for reconfiguration tasks.

In order to simplify the description of the formulation, it is useful to define the following sets derived from $map_{t,i,c}$:

$TIC :=$ set of triplets (t, i, c) where $t \in T, i \in I, c \in C$ such that $map_{t,i,c} = 1$;
$TI :=$ set of couples (t, i) where $t \in T, i \in I$ such that $\exists c \in C : map_{t,i,c} = 1$;
$TC :=$ set of couples (t, c) where $t \in T, c \in C$ such that $\exists i \in I : map_{t,i,c} = 1$.

Moreover, starting from the definition already presented, we are able to compute additional sets that will be exploited to give a better characterization of the model variables:

$CP :=$ component precedence set: contains all the couples of tasks $(t1, t2) : t1, t2 \in T$ such that it is possible to schedule $t2$ right after $t1$ on the same hardware component;
$OT :=$ overlapping tasks set: contains all the couples of tasks $(t1, t2) : t1, t2 \in AT$ such that there exists a schedule in which $t1$ and $t2$ overlap in time;
$CT :=$ compatible tasks set: contains all the couples of tasks $(t1, t2) : t1, t2 \in T$ such that both tasks have at least one common hardware implementation ($\exists i \in I^h : (t1, i), (t2, i) \in TI$).

More formally the sets can be computed as:

$$
\begin{aligned}
CP = \{&(t1,t2) \mid t1,t2 \in T \wedge t1 \neq t2 \wedge (t2,t1) \notin P^+ \wedge \\
&(\exists c \in C^h \mid (t1,c),(t2,c) \in TC)\} \\
OT = \{&(t1,t2) \mid t1,t2 \in AT \wedge t1 \neq t2 \wedge \sim (t1,t2 \in RT) \wedge \\
&(t1,t2) \notin P^+ \wedge (t2,t1) \notin P^+\} \\
CT = \{&(t1,t2) \mid t1,t2 \in T \wedge t1 \neq t2 \wedge \\
&(\exists i \in I^h, c \in C^h \mid (t1,i,c),(t2,i,c) \in TIC)\}
\end{aligned} \tag{1}
$$

Notice for the definition of OT we take into account a single reconfigurator, so that no two reconfiguration tasks can overlap in time by definition.

B. Variables identification

By using the sets and parameters described in the previous section, we are now able to introduce the variables required for the MILP model:

- $b_t :=$ $\forall t \in AT$: real variable in the range $[0, T_{max}]$ defining the begin time of task t;
- $e_t :=$ $\forall t \in AT$: real variable in the range $[0, T_{max}]$ defining the end time of task t;
- $mic_{t,i,c} := \forall (t,i,c) \in TIC$: binary variable set to 1 if task t is mapped to component c with implementation i;
- $mi_{t,i} := \forall (t,i) \in TI$: binary variable set to 1 if task t is assigned to implementation i;
- $mc_{t,c} := \forall (t,c) \in TC$: binary variable set to 1 if task t is mapped to component c;
- $oc_{c,r} := \forall c \in C^h, r \in R$: real non negative variable (≥ 0) defining the amount of resources of type r needed by hardware component c;
- $cp_{t1,t2} := \forall (t1,t2) \in CP$: binary variable set to 1 if task $t2$ is executed right after task $t1$ on the same hardware component;
- $cft_{t,c} := \forall (t,c) \in TC : c \in C^h$: binary variable set to 1 if task t is the first task executed on hardware component c;
- $rtt_{rt,t} := \forall t \in T, rt \in RT$: binary variable set to 1 if task t requires reconfiguration rt prior to its execution;
- $rtc_{rt,c} := \forall rt \in RT, c \in C^h$: binary variable set to 1 if reconfiguration task rt is performed on hardware component c;
- $bitc_c := \forall c \in C^h$: real non negative variable (≥ 0) defining the bitstream size for hardware component c;
- $ba_{t1,t2} := \forall (t1,t2) \in OT$: binary variable set to 1 if task $t1$ begins after the beginning of task $t2$ or at the same time of $t2$;
- $bb_{t1,t2} := \forall (t1,t2) \in OT$: binary variable set to 1 if task $t1$ begins before the end of task $t2$ or at the end of $t2$;
- $bo_{t1,t2} := \forall (t1,t2) \in OT$: real variable in the range $[0,1]$ set to 1 if the beginning of task $t1$ overlaps in time with task $t2$ (i.e. task $t1$ begins when $t2$ is in execution);
- $mibo_{t1,t2,i} := \forall (t1,t2) \in OT, (t2,i) \in TI$: real variable in the range $[0,1]$ set to 1 if $bo_{t1,t2} = 1$ and $mi_{t2,i} = 1$.

All the time intervals considered in the model are closed on the left and open on the right, meaning that instant b_t represents the first time instant in which task t is in execution, while instant e_t represents the first instant in time right after the end of the execution of task t. As an example, if the time domain is discretized into clock cycles, we have $T_\epsilon = 1$ while the computation of a task t with $b_t = 2$ and $e_t = 5$ is performed during clock cycles 2, 3 and 4. Moreover, in order to speed up the execution time of the MILP solver, variables $oc_{c,r}$, $bitc_c$, $bo_{t1,t2}$, $mibo_{t1,t2,i}$ are declared as real even thought their values should be integer. This can be done without changing the semantics of the model thanks to the constraints of the next sections that relate these variables to the other integer variables of the formulation.

C. Semantic constraints

Within this section we define all the constraints of the MILP formulation that are required to guarantee the semantics of the model variables

Guarantees the semantics for bb, ba and bo:

$$
\begin{aligned}
&\forall (t1,t2) \in OT : \\
&\quad b_{t1} \geq e_{t2} - bb_{t1,t2} \cdot T_{max} \\
&\quad b_{t1} \leq b_{t2} - T_\epsilon + ba_{t1,t2} \cdot T_{max} \\
&\quad bo_{t1,t2} \geq ba_{t1,t2} + bb_{t1,t2} - 1
\end{aligned} \tag{2}
$$

Ensures that a task is mapped exactly on one implementation and one component:

$$
\forall t \in T : \sum_{(t,i,c) \in TIC} mic_{t,i,c} = 1 \tag{3}
$$

Computes mi and mc from mic:

$$
\begin{aligned}
\forall (t,i) \in TI : mi_{t,i} &= \sum_{(t,i,c) \in TIC} mic_{t,i,c} \\
\forall (t,c) \in TC : mc_{t,c} &= \sum_{(t,i,c) \in TIC} mic_{t,i,c}
\end{aligned} \tag{4}
$$

Computes the end of a task with respect to the selected implementation:

$$
\forall t \in T : e_t = b_t + \sum_{(t,i) \in TI} mic_{t,i} \cdot time_i \tag{5}
$$

Ensures that there is at most one first task executed on a hardware component:

$$
\forall c \in C^h : \sum_{(t,c) \in TC} cft_{t,c} \leq 1 \tag{6}
$$

Relates cft with mc (i.e. if a task is the first on a hardware component, it must be mapped to that component):

$$
\forall (t,c) \in TC \mid c \in C^h : cft_{t,c} \leq mc_{t,c} \tag{7}
$$

Relates cp with mc (i.e. $cp_{t1,t2} = 0$ if $t1$ and $t2$ are mapped on different components):

$$
\begin{aligned}
&\forall (t1,t2) \in CP, \forall c1 \in C : \\
&mc_{t1,c1} + \sum_{(t2,c2) \in TC : c2 \neq c} mc_{t2,c2} + cp_{t1,t2} \leq 2
\end{aligned} \tag{8}
$$

Relates cft with cp and mc (i.e. if a task is mapped to a hardware component then it is the first task or another task is executed before it):

$$\forall(t,c) \in TC \mid c \in C^h : \\ mc_{t,c} \leq cft_{t,c} + \sum_{(t2,t)\in CP} cp_{t2,t} \tag{9}$$

Relates cp with the scheduling of the tasks:

$$\forall(t1,t2) \in CP : \\ b_{t2} \geq e_{t1} - (1 - cp_{t1,t2}) \cdot T_{max} \tag{10}$$

Ensures that a hardware component occupies not less than the resources needed by the tasks mapped on it:

$$\forall c \in C^h, \forall r \in R, \forall t \in T : \\ oc_{r,c} \geq \sum_{(t,i,c)\in TIC} mic_{t,i,c} \cdot res_{i,r} \tag{11}$$

Computes the bitstream size of a hardware component:

$$\forall c \in C^h : bitc_c = \sum_{r\in R} oc_{c,r} \cdot bit_r \tag{12}$$

Ensures that a task requires at most one reconfiguration and that a reconfiguration configure no more than one task:

$$\forall rt \in RT : \sum_{t\in T} rtt_{rt,t} \leq 1 \\ \forall t \in T : \sum_{rt\in RT} rtt_{rt,t} \leq 1 \tag{13}$$

Guarantees the semantic of rtc:

$$\forall rt \in RT, \forall c \in C^h, \forall t \in T \mid (t,c) \in TC : \\ rtc_{rt,c} \geq rtt_{rt,t} + mc_{t,c} - 1 \tag{14}$$

Ensures that a reconfiguration task ends always after its beginning:

$$\forall rt \in RT : e_{rt} \geq b_{rt} \tag{15}$$

Ensures that if a reconfiguration task is performed on a component, the reconfiguration cannot last less than required:

$$\forall rt \in RT, \forall c \in C^h : \\ e_{rt} \geq b_{rt} + (bitc_c - (1 - rtc_{rt,c}) \cdot bit_{max}) \cdot T_{rec} \tag{16}$$

D. Problem constraints

having guaranteed the semantics of the variables, we are able to define the constraints tightly related to the problem.

Ensures the dependencies among the tasks:

$$\forall(t1,t2) \in P : b_{t2} \geq e_{t1} \tag{17}$$

Avoids overlap among tasks mapped on the same component (we exploit the fact that if $bo_{t1,t2} = bo_{t2,t2} = 0$ there is no overlapping among tasks $t1, t2 \in OT$):

$$\forall(t1,t2) \in OT, \forall c \in C \mid (t1,c),(t2,c) \in TC : \\ bo_{t1,t2} + mc_{t1,c} + mc_{t2,c} \leq 2 \tag{18}$$

Ensures non overlapping also with respect to cp (i.e. given a task t, there is at most one previous task and one subsequent task on the same hardware component):

$$\forall t \in T : \\ \sum_{(t,t2)\in CP} cp_{t,t2} \leq 1 \quad \sum_{(t2,t)\in CP} cp_{t2,t} \leq 1 \tag{19}$$

Avoids overlapping between the potential reconfiguration tasks by enforcing a sequential order (to state this inequality we assume the reconfiguration tasks assigned to unique natural numbers in the interval $[1, |T| - 1]$):

$$\forall rt \in RT \mid rt > 1 : b_{rt} \geq e_{rt-1} \tag{20}$$

Ensures that the hardware components do not exceed the resources provided by the FPGA:

$$\forall r \in R : \sum_{c\in C^h} oc_{c,r} \leq maxRes_r \tag{21}$$

Ensures that between two subsequent tasks $t1, t2 \in T$ mapped on the same hardware component with different implementations a reconfiguration must be performed to configure task $t2$:

$$\forall(t1,t2) \in CP, \forall i1 \in I^h \mid (t1,i1) \in TI \wedge (t1,t2) \in CT : \\ \sum_{rt\in RT} rtt_{rt,t2} \geq cp_{t1,t2} + mi_{t1,i1} + \sum_{\substack{(t2,i2)\in TI \mid \\ i2\in I^h \wedge i1 \neq i2}} mi_{t2,i2} - 2 \\ \forall t \in T : \\ \sum_{rt\in RT} rtt_{rt,t} \geq \sum_{(t2,t)\in CP:(t,t2)\notin CT} cp_{t2,t} \tag{22}$$

Guarantees that a reconfiguration between tasks $t1 \in T$ and $t2 \in T$ is executed after $t1$ and before $t2$:

$$\forall rt \in RT, \forall t \in T : \\ b_t \geq e_{rt} - (1 - rtt_{rt,t}) \cdot T_{max} \\ \forall rt \in RT, \forall(t,t2) \in CP : \\ e_t \leq b_{rt} + (2 - rtt_{rt,t2} - cp_{t,t2}) \cdot T_{max} \tag{23}$$

Notice that the proposed model currently does not directly consider the delay due to communication among tasks. However, Equation 17 can be easily modified to take into account a fixed $\lambda_{t1,t2}$ communication time among tasks $t1$ and $t2$ that is not dependent on the selected implementations:

$$\forall(t1,t2) \in P : b_{t2} \geq e_{t1} + \lambda_{t1,t2} \tag{24}$$

E. Objective function

Using the variables and the parameters previously defined we are able to compute and optimize the following three different metrics within the MILP model:

Execution time (T_{cost}): The overall execution time needed to complete the computation of all the tasks of the schedule including reconfiguration tasks;.

Peak power (P_{cost}): The estimated peak power reached by the schedule, this value is computed considering the

maximum overall power consumption reached within a single time unit;

Energy consumption (E_{cost}): The estimated energy consumption for the schedule, it is computed considering the specific implementation selected for each task and the energy needed for all the reconfigurations.

In order to define the metrics within the model, we consider T_{cost}, P_{cost} and E_{cost} as new non negative (≥ 0) real variables. Since the objective is to minimize a suitable combination of these metrics, it is enough to provide lower bounds for variables T_{cost}, P_{cost} and E_{cost}.

The lower bound for the execution time is given by the following constraint:

$$\forall t \in AT : T_{cost} \geq e_t \tag{25}$$

In order to compute lower bounds for the peak power we do not need to consider all the time instants within the schedule, indeed, the peak power varies only when a task begins or ends its execution. Furthermore, when a tasks ends its execution, the peak power cannot increase and this allows us to take into account only the time instants in which a task begin its execution. For the beginning of each task $t \in AT$ we consider the total power of all the tasks that are in execution at time b_t (i.e. all tasks $t2 \in AT$ such that $bo_{t,t2} = 1$). The following are the lower bounds for P_{cost} considering both time instants derived from application tasks and reconfiguration tasks (variable rtt is used to remove not required reconfigurations):

$$\begin{aligned} \forall t \in T : & P_{cost} \geq \sum_{(t,i)\in TI} mi_{t,i} \cdot power_i + powRT_t + powT_t \\ \forall rt \in RT : & P_{cost} \geq \sum_{t\in T} rtt_{rt,t} \cdot P_{rec} + powT_{rt} \end{aligned} \tag{26}$$

Where:

$$\begin{aligned} powRT_t &= \sum_{\substack{rt\in RT, t2\in T| \\ (t,rt)\in OT}} rtt_{rt,t2} \cdot P_{rec} \\ powT_t &= \sum_{\substack{t2\in T,(t2,i)\in TI| \\ (t,t2)\in OT}} mibo_{t,t2,i} \cdot power_i \end{aligned} \tag{27}$$

On the other hand, regarding the energy consumption, we are able to compute an exact value as:

$$E_{cost} = \sum_{(t,i)\in TI} mi_{t,i} \cdot time_i \cdot power_i + \sum_{rt\in RT} (e_{rt} - b_{rt}) \cdot P_{rec} \tag{28}$$

Notice that for a reconfiguration rt that is not required, the solver is allowed to set $b_{rt} = e_{rt}$ so that it does not impact on the energy cost. Furthermore, static power and static energy consumption can be easily taken into account by augmenting Equations 26 and 28 with a constant term and a linear time term respectively. Overall, a possible objective function for the problem can be obtained with a linear combination of T_{cost}, P_{cost} and E_{cost}:

$$\min \left\{ q_1 \cdot \frac{T_{cost}}{T_{max}} + q_2 \cdot \frac{P_{cost}}{P_{max}} + q_3 \cdot \frac{E_{cost}}{E_{max}} \right\} \tag{29}$$

Where T_{max}, P_{max} and E_{max} are normalization terms representing the maximum value that T_{cost}, P_{cost} and E_{cost} can achieve respectively. While q_1, q_2 and q_3 are weights that can be set according to the designer preferences.

F. Cutting planes

Within this final section we refine the formulation by introducing additional constraints whose goal is to provide a better description of the convex hull and to improve the numerical properties of the model.

In the previous section, we used Equation (18) to avoid overlapping among tasks on the same component. This constraint depends on variable bo that in turns depends on bb and ba. The coherence of variables bb and ba is guaranteed by Equation (2). When the T_{max} parameter is too big, the solver may fail to attribute the correct values to bb and ba due to numerical errors, so that overlapping is not correctly detected. To overcome this issue we introduce an additional non overlapping constraint that binds variables bb and ba directly to variables mc while improving at the same time the description of the model:

$$\begin{aligned} \forall (t1,t2) \in OT, \forall c \in C \mid (t1,c) \in TC \wedge (t2,c) \in TC : \\ mc_{t1,c} + mc_{t2,c} - 1 \leq bb_{t1,t2} + bb_{t2,t1} \\ mc_{t1,c} + mc_{t2,c} - 1 \leq ba_{t1,t2} + ba_{t2,t1} \end{aligned} \tag{30}$$

The formulation can be further improved by removing symmetries that depend on the application domain. If we consider an optimal schedule, another solution with the same quality can be easily obtained by applying a permutation of the indexes of the hardware and the software components. To remove this high amount of equivalent solutions we can simply add constraints that prevent the mapping of some of the tasks to some of the components. If we consider both the hardware and software components and the tasks assigned to increasing numbers starting from 1, the symmetries can be removed as follows:

$$\begin{aligned} \forall c \in C, \forall t \in T \mid t \leq c \wedge (t,c) \in TC : \\ mc_{t,c} = 0 \end{aligned} \tag{31}$$