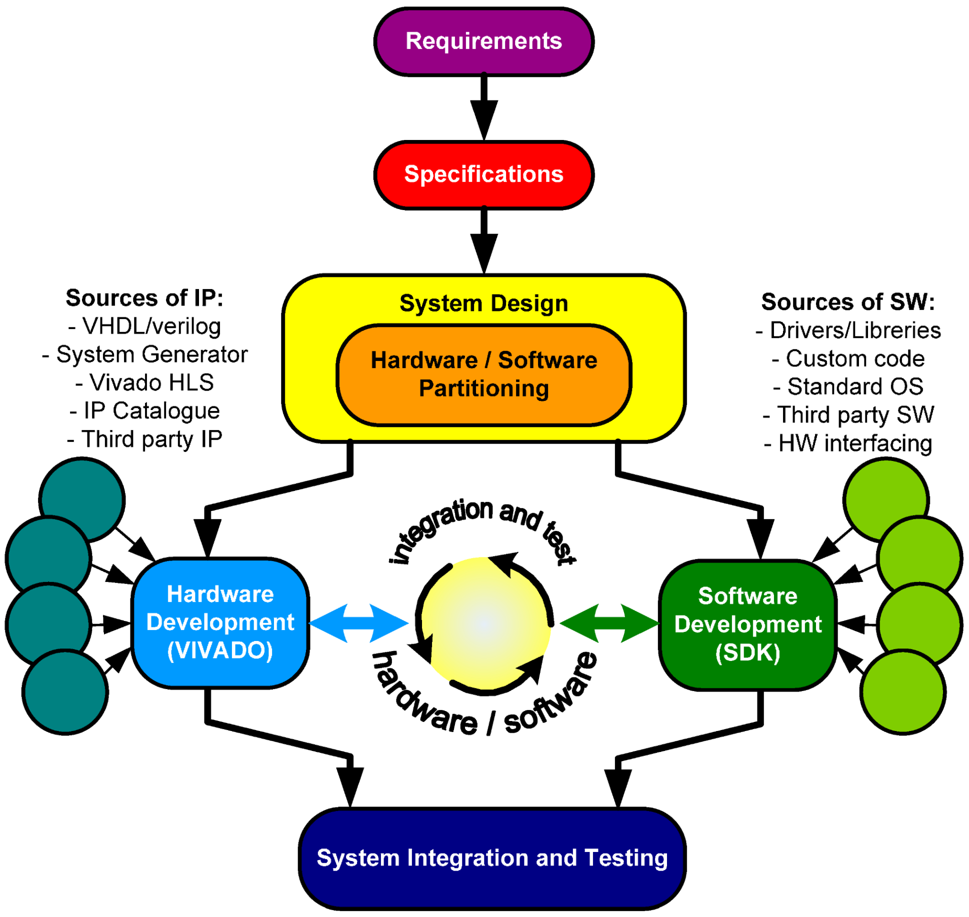

AIFA Interim Report

Topic : Heuristic Search Methods for Functional partitioning in Hardware-Software codesign

27.10.2021

Group members

Abhisek Mohanty (18EC3AI20)

Aishik Mandal (18EC3AI21)

Hardhik Mohanty (18EE3AI26)

Samarth (18EC35051)

Problem Statement

I. Problem statement

Task partitioning and task scheduling are required in many applications, for instance codesign systems, parallel processor systems, and reconfigurable systems. Sub-tasks extracted from the input description should be implemented in the right place (using the partitioner) at the right time (using the scheduler). It is well known that such scheduling and partitioning problems are NP-complete and are therefore intractable. Furthermore, in the case of reconfigurable hardware, such problems are considered NP-hard. Optimization techniques based on heuristic methods are generally employed to explore the search space so that feasible and near-optimal solutions can be obtained.

Here we focus on algorithms involving heuristic search to solve hardware software partitioning problems. Although heuristic methods do not guarantee the optimum point, they can produce acceptable solutions within a reasonable amount of time.Common heuristic-based algorithms include Depth First Branch & Bound (DFBB), simulated annealing (SA), and genetic algorithm (GA). We are planning to model the problem as a CSP and then apply Genetic Algorithm (GA) and Simulated annealing(SA) to get the optimal scheduling.

II. Formal Problem statement

Given : A task graph with each node T_i has hardware area cost h_i and software execution time cost s_i

The binary decision variable is y_i.

y_i =1, if T_i is implemented by Software, else if y_i=0, then it is implemented by hardware.

Optimization objective :

$$\sum_i (1 - y_i)h_i$$

Subject to constraint:

$$\sum_i y_i.s_i \leq D$$

Where, D denotes the total Software resources available

AI modelling

The problem we are trying to solve has the constraint of execution time and we can not go beyond certain timing constraints for scheduling. At the same time we are also trying to minimize the hardware resources used in terms of its total area in order to reduce the size of our system.

Given the above scenario we can model the problem as a CSP and use the CSP solver. We can also use search to get a solution. But the optimal solution is to search the whole state space using some advanced search technique. This can be time as well as resource consuming. Thus we use a Heuristic based Search. It is useful for pruning when overhead for optimal solution is unacceptable. Depending upon the heuristic function we can produce solutions close to the optimal solution.

Solution Approach

Programming Language: Python

Branch & Bound(BB)

- State space tree expansion starts at a state when all tasks are unallocated.
- Make a move by choosing one of the tasks
- At a given step, we have two move options
  - Allocate task for hardware execution
  - Allocate task for software execution
  - Branch on all possible options on the state reached through the move iff,
    - Current best solution cost < Upper Bound at that state.
    - Otherwise, prune or kill the search of the subsequent sub-tree.

Depth First Branch & Bound (DFBB)

- Initially, the algorithm maintains a stack (LIFO structure) for newly generated states of the state space tree.
- A goal state holds a valid allocation (HW/SW) for every task.
- The optimal goal state is the one that has the minimal HW area cost.
- There is an upper bound function f(n) in every state: f(n) = g(n) + h(n)
  - g(n) = Actual HW cost of tasks already allocated to SW
  - g'(n) = Actual SW execution time cost at current state

- h(n) = Upper bound on estimated HW cost (of software tasks) of reaching goal state from present state

$$h(n) = \sum_{j=l1}^{lk} h_j + (h_{lk+1}/s_{lk+1})^* \left(D - g'(n) - \sum_{j=l1}^{lk} s_j \right)$$

such that the following holds,

$$\sum_{j=l1}^{lk} s_j \leq D - g^{'}(n) \leq \sum_{j=l1}^{lk+1} s_j$$

References

1. An Effective Heuristic-Based Approach for Partitioning
2. Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware–Software Codesign
3. What is hardware/software partitioning?
4. Hardware-software partitioning in embedded system design