High-Level Synthesis - ASAP and ALAP Scheduling

1st Kristina Bala
Hochschule Hamm-Lippstadt
Electronic Engineering
4th Semester
kristina.bala@stud.hshl.de

***Abstract*—**
***Index Terms*—**

I. First References

[1]()
[2]()

II. Introduction

Synthesis and optimization of circuits at the architectural level.

Techniques for transforming an abstract model of circuit behavior into data path and control unit.

III. Architectural-Level Synthesis and Optimization

Synthesis and optimization of circuits at the architectural level.

Techniques for transforming an abstract model of circuit behavior into data path and control unit.

Data path - interconnection of resources (implementing arithmetic or logic functions) whose execution times and I/O data are determined by the control unit according to a SCHEDULE.

Architectural synthesis:
- Structural view of the circuit, its data path
- A logic-level specification of its control unit

Circuit implementation:
- Area
- Cycle-time (clock period)
- Latency (i.e. nr of cycles to perform all operations)
- Throughput (i.e. the computation rate)

Circuit specifications for architectural synthesis

1. Behavioral-level circuit models
2. Details about the recourses being used
3. Constraints

IV. High Level Synthesis Design Steps

Key concepts Starting from the high-level description of an application, an RTL component library, and specific design constraints, an HLS tool executes the following tasks (see Figure 1):
1. compiles the specification,
2. allocates hardware resources (functional units, storage components, buses, and so on),
3. schedules the operations to clock cycles,
4. binds the operations to functional units,
5. binds variables to storage elements,
6. binds transfers to buses, and
7. generates the RTL architecture. (Reference here)

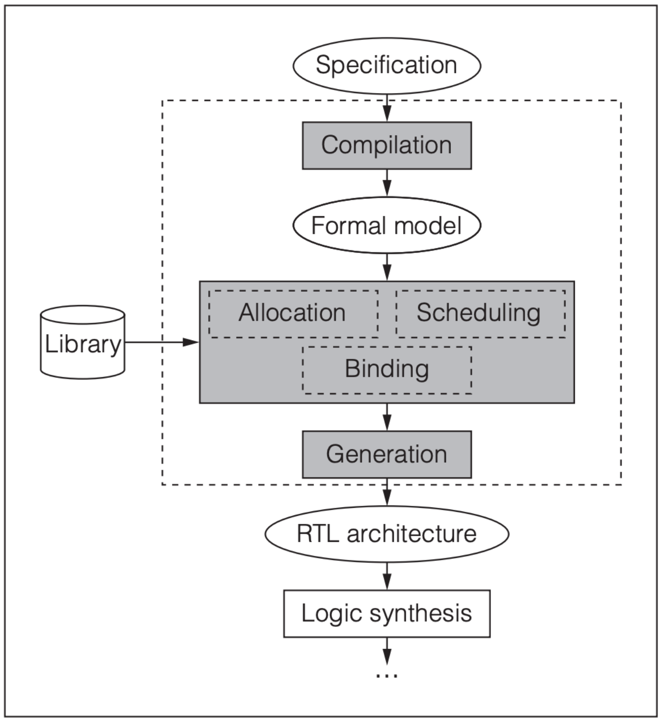

Fig. 1. HLS Design Steps

V. Scheduling

Scheduling is a very important problem in architectural synthesis.

-the task of determining the start of times, subject to the precedence constraints specified by the sequencing graph

Sequencing graph – prescribes only the dependencies among operations

Scheduling of sequencing graph:

- determines the precise time of each task
- concurrency of resulting implementation
- affects the area of implementation (the max nr of concurrent operations of any given type at any step of the schedule is a lower bound on the number of required hardware resource)

A. *Scheduling without resource constraints*

applied when:

- dedicated resources are used

  when operators differ in their types

  or when their cost is marginal when compared to that of steering logic, registers. Wiring and control
- resource binding is done prior to scheduling and resource conflicts are solved by serializing the operations that share the same resource.

  The area cost of implementation is defined before and independently from the scheduling step

Used to:

-derive bounds on latency for constrained problems

the minimum latency of a schedule under some resource constraint is at least as large as the latency computed with unlimited resources

VI. Allocation

To satisfy the design constraints, allocation is needed to define the type and number of hardware resources (for instance, functional units, storage, or connectivity components). While it depends on the HLS tool, some other components like buses or point-to-point connections among components for the connectivity part, may be added during scheduling and binding tasks. For each operation in the specification model, it must be selected at least one component from the RTL component library which includes their characteristics (such as area, delay, and power) and its metrics to be used by other synthesis tasks.

VII. Binding

A storage unit must be bound to each variable that carries values across cycles. If several variables have nonoverlapping or mutually exclusive lifetimes, they could be bound to the same storage units. On the other hand, each operation in the specification model must be bound as well to one of the functional units, that is capable of executing that specific operation. Therefore, the binding algorithm must optimize the selection where several units have such capability. Storage and functionalunit binding also depend on connectivity binding, which requires that each transfer from component to component be bound to a connection unit such as a bus or a multiplexer. [3] In an ideal case, HLS is able to estimate the connectivity delay and area as early as possible, in order to achieve a better optimization of the design. Alternatively, the complete architecture can be specified during allocation, which can result to initial floorplanning that can be used during binding and scheduling.

VIII. Unconstrained Scheduling: The ASAP Scheduling Algorithm

- The start time for each operation is the least one allowed by the dependencies
- Optimize the overall latency of the computation without caring about the number of resources required
- Finding the longest path between each operation and the source node
- Starting each operation in a CDFG (Control Data Flow Graph) as soon as its predecessors have completed

IX. Latency-Constrained Scheduling: The ALAP Scheduling Algorithm

- Schedule operations at the latest opportunity
- Seeking the longest path between each operation and the end or “sink” node.
- Schedule start times can be derived by subtracting the longest path time from the desired overall latency constraint

X. Application Example

XI. Outlook

A. *Drawbacks*

B. *Benefits*

Acknowledgment