Skysong Innovations
Advancing the Arizona State University Knowledge Enterprise

1475 N. Scottsdale Road, Suite 200
Scottsdale, AZ 85287-3538
Phone: 480 884 1996 Fax: 480 884 1984

Case ID:M21-079P^
Published: 9/21/2021

Inventors

Mahesh Balasubramanian

Aviral Shrivastava

Contact

Shen Yan
shen.yan@skysonginnovations.com

Improved Mapping of Computational Loops on Reconfigurable Architectures

Today there are numerous devices that collect, process, and communicate data from multiple sources such as the internet, cyber-physical and autonomous systems, and sensor networks. Extracting intelligent and actionable information from this data—whether done by machine learning or otherwise—is extremely compute-intensive and often limited by power, thermal, and other resource constraints. Execution efficiency of these functionalities can be achieved by using application-specific integrated circuits (ASICs). However, they suffer from high production costs, and quickly become obsolete as applications and algorithms evolve. Another promising alternative is field programmable gate arrays (FPGAs), which lose efficiency in providing bit-level configurability, that is essential for their primary purpose—prototyping. Coarse-grained reconfigurable architectures (CGRAs) provide a balanced middle ground with coarse-grain configurability (word- and arithmetic-operator-level), with minimal loss in power efficiency relative to ASICs.

The acceleration achieved by CGRAs relies on the efficient mapping of the compute-intensive loops by the CGRA compiler onto the CGRA architecture. The CGRA mapping problem is performed in a two-step process, namely scheduling and mapping. The scheduling algorithm allocates timeslots to nodes of a data flow graph (DFG), and the mapping algorithm maps the scheduled nodes onto the processing elements (PEs) of the CGRA. On a mapping failure, the initiation interval (II) is increased and a new schedule is obtained for the increased II. Most previous mapping techniques use iterative modulo scheduling (IMS) to find a schedule for a given II. Since IMS generates a resource-constrained ASAP (as soon as possible) scheduling even with increased II, it tends to generate a similar schedule that is not mappable. Therefore, IMS does not fully explore the schedule space. Researchers at Arizona State University have developed a scheduling and mapping technique that addresses the mapping shortcomings of iterative modulo scheduling (IMS). Specifically, this involves generating random modulo schedules within the schedule space, thereby creating different modulo schedules at a given and increased initiation interval (II).

Both the Resource Constrained As Soon As Possible (RC_ASAP) and Resource Constrained As Late As Possible (RC_ALAP) schedules are generated for all the nodes of the data flow graph (DFG), similar to the concept of mobility used in high-level synthesis (HLS). Then the algorithm chooses a random scheduling time between RC_ASAP and RC_ALAP for each node. As a result, every time a "new" schedule is obtained, the schedule space is effectively explored. A novel conservative feasibility test is subsequently performed, which ensures mappability of the obtained schedule even upon addition of the new routing nodes. Among the

24 performance-critical loops (that account for more than 7% of execution time of the application) from MiBench, Rodinia, and Parboil, this innovative approach was able to map all the loops for various CGRA sizes ranging from 4×4 to 8×8, and achieved a comparable II for the loops which were mappable by RAMP.

Related Publication: CRIMSON: Compute-intensive loop acceleration by Randomized Iterative Modulo Scheduling and Optimized Mapping on CGRAsPotential Applications:

- Acceleration of compute-intensive loops
- Course-grained reconfigurable arrays/architectures
- Cyber-physical systems
- Autonomous systems

Benefits and Advantages:

- Detected unmappable schedules are eliminated and their associated mapping algorithms are not invoked
  - Saves computing time and accelerates computing rate by reducing unnecessary computing times of unmappable schedules