Early-stage Automated Identification Tool for Shared Accelerators

Parnian Mokri
Tufts University
Medford, Massachusetts 02155
Email: parnian.mokri@tufts.edu

Mark Hempstead
Tufts University
Medford, Massachusetts 02155
Email: mark.hempstead@tufts.edu

The use of application-specific accelerators to improve systems' energy-efficiency and performance is becoming more prevalent. To overcome the tight area budget on embedded systems we propose an early detection tool that complements existing High-level Synthesis tools by identifying computationally similar synthesizable kernels that are used to build Shared Accelerators (SAs). SAs are specialized hardware accelerators that execute very different software kernels but share the common hardware functions between them. SAs can provide increased coverage if similarities between the dataflow and control flow of seemingly very different workloads are detected. Existing methods use either dynamic traces or analyze register transfer level (RTL) implementations to find these similarities which requires deep knowledge of RTL and the time-consuming RTL design process.

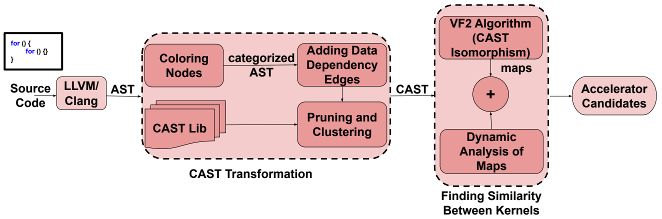


Fig. 1. Block diagram of the ReconfAST methodology.

This work leverages abstract syntax trees (ASTs) generated from LLVM's-clang to discover similar kernels among workloads. ASTs are compact, unlike control and dataflow representations, but contain extra syntax and variable node ordering that complicates workload comparison. As shown in Figure 1 our methodology, ReconfAST, transforms the AST into a new clustered AST (CAST) representation that further removes unneeded nodes and uses a flexible tree-traversal and regular expression matching scheme to detect and group common node patterns. ReconfAST transforms ASTs into a hardware implementable tree by removing whitespace nodes to remove differences that are resulted from coding style. We run a dynamic analysis of these static structural similarities, to further refine SA candidates by making sure these maps represent hot code. Finally, we prune the candidates based on their static data dependency class. This step will remove cases when a variable inside the acceleration candidate depends on outside variables. In addition, the tool warns the user in cases where many data dependencies are found inside the acceleration candidate since that would limit the ability of HLS tools to use common hardware optimizations to improve performance and energy efficiency. Figure 2 shows a simplified example of a system with accelerators for two MachSuite benchmarks, *Stencil2D* and *Viterbi*. Figure 3 shows that the common source code between stncil2d and vterbi was 94% of stencil2D hot-code. Therefore, a common accelerator can accelerate both workloads.

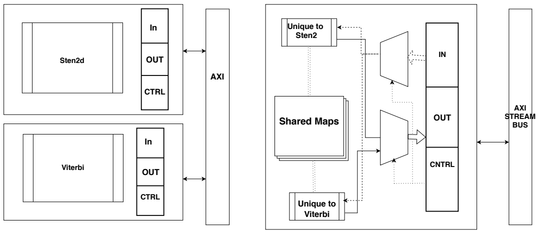
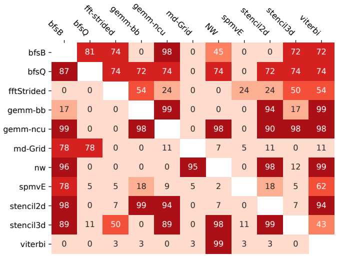


Fig. 2. The two Dedicated Accelerators (on the left) are replaced by one Shared Accelerator (on the right), increasing accelerator's coverage and saving area.



	bfsB	bfsQ	fft-strided	gemm-bb	gemm-ncu	md-Grid	NW	spmvE	stencil2d	stencil3d	viterbi
bfsB		81	74	0	98	0	45	0	0	72	72
bfsQ	87		74	72	74	0	74	0	72	74	74
fftStrided	0	0		54	24	0	0	24	24	50	54
gemm-bb	17	0	0		99	0	0	0	94	17	99
gemm-ncu	99	0	0	98		0	98	0	90	98	98
md-Grid	78	78	0	0	11		7	5	11	0	11
nw	96	0	0	0	0	95		0	98	12	99
spmvE	78	5	5	18	9	5	2		18	5	62
stencil2d	98	0	7	99	94	0	7	0		7	94
stencil3d	89	11	50	0	89	0	98	11	99		43
viterbi	0	0	3	3	0	3	99	3	3	0	

Fig. 3. Maximum Dynamic Coverage (percentage of total execution time) measured of the matching (isomorphic) sub-graphs found between the CASTs of each workload.

The presence of data dependencies, the cost of reconfiguration, and the difference between the size of accelerators affect the efficiency of SAs. We have designed over 700 of these accelerators using Vivado_HLS. A good Shared Accelerator, on FPGAs has comparable speedup to dedicated accelerators and reduces the resources required for FPGA implementations: 37% FFs, 16% DSPs, and 10% on LUTs on average.