QuickDough: A Rapid FPGA Accelerator Design Methodology Using Soft Coarse-Grained Reconfigurable Array Overlay

Cheng Liu, Colin Yu Lin, Hayden Kwok-Hay So
liucheng@eee.hku.hk, linyu@eee.hku.hk, hso@eee.hku.hk

Background and Motivation

The design productivity of developing an FPGA accelerator remains a major obstacle that hinders it from wide adoption. FPGA overlay which can be parametric HDL, pre-synthesized or pre-implemented circuit is a promising technique to tackle the design productivity challenge. It is beneficial to FPGA design productivity mainly from the following three aspects.

- It usually uses DFG or even high level language program as input, and thus raises the abstraction level of design entry.
- It typically has coarser configuration granularity and simplifies the compilation. Therefore, it helps to reduce the compilation time.
- It can be reused across the FPGA devices and parts, which indicates more convenient design reuse and portability.

SCGRA Based Accelerator

Figure 1 shows the structure of the soft coarse-grained reconfigurable array (SCGRA) based FPGA accelerator. As the implementation of the SCGRA has significant impact on the overall performance, one PE of the SCGRA is further presented in Figure 2.

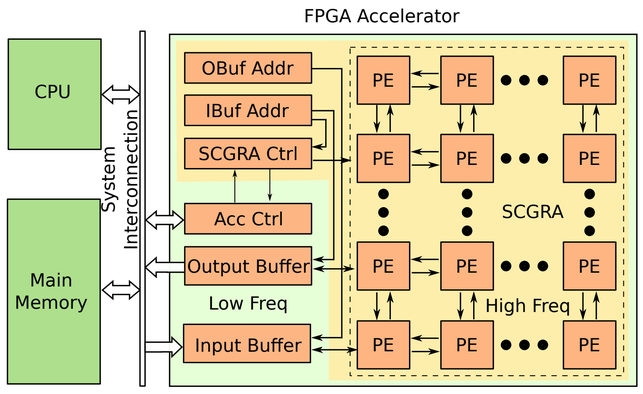

Figure 1: SCGRA Based FPGA Accelerator

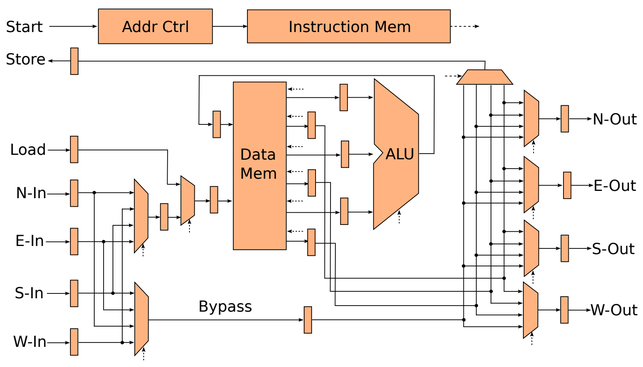

Figure 2: PE Structure

Conclusion

QuickDough using the SCGRA overlay as an intermediate compilation step simplifies the lengthy FPGA implementation. The time compiling from high level language program to CPU+FPGA system can be reduced by around two magnitudes. In addition, implementation with higher clock frequency resulting from the highly regular structure of SCGRA in combination with effective operation scheduling provides competitive performance compared to a commercial HLS based design.

QuickDough

QuickDough an SCGRA based FPGA accelerator design methodology as shown in Figure 3 is proposed to compile an application to a hybrid CPU+FPGA system. It has the compute intensive kernels scheduled to the SCGRA overlay and the rest compiled to CPU.

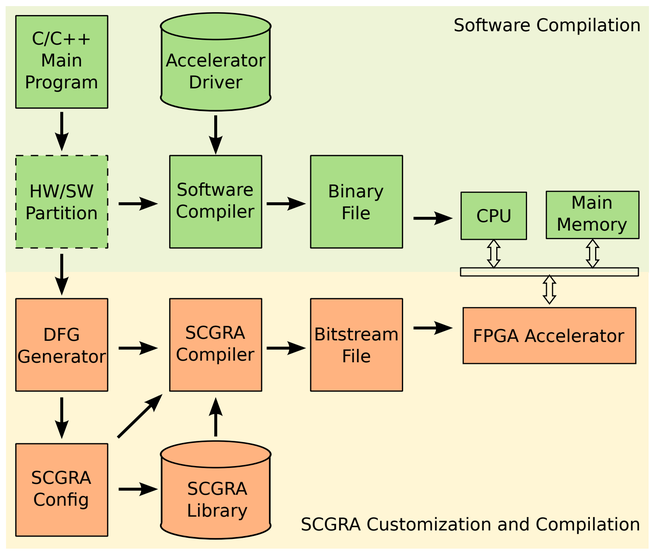

Figure 3: QuickDough

Experiment-1

Design productivity, performance, implementation frequency and scalability of both Vivado HLS basd design methodology and QuickDough are presented. Design productivity comparison is shown in Figure 4 and 5. End-to-end performance is compared in Figure 6. Kernel simulation performance and SCGRA implementation frequency are compared in Figure 7 and 8 respectively. Implementation scalability of the SCGRA overlay is presented in Figure 9.

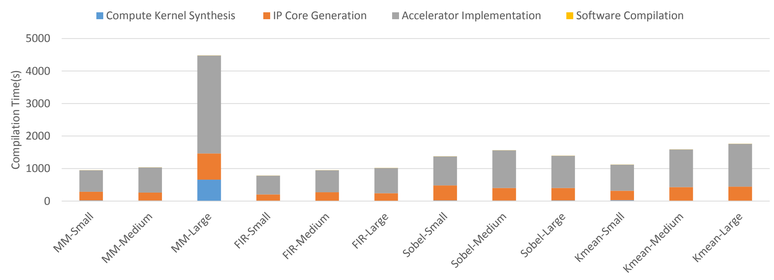

Figure 4: Compilation Time Using Vivado HLS Based Design Methodology

Experiment-2

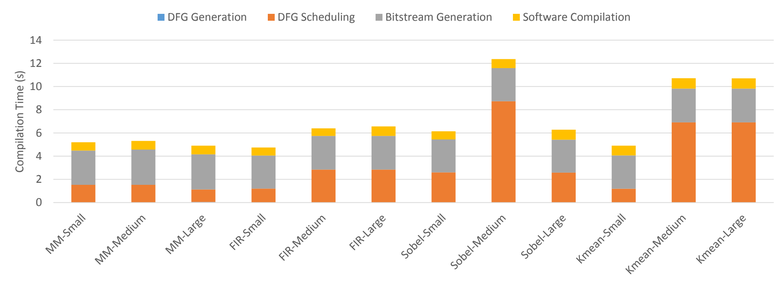

Figure 5: Compilation Time Using QuickDough

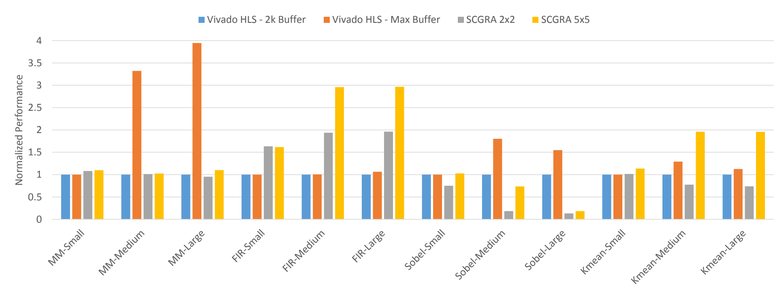

Figure 6: Normalized Overall Performance

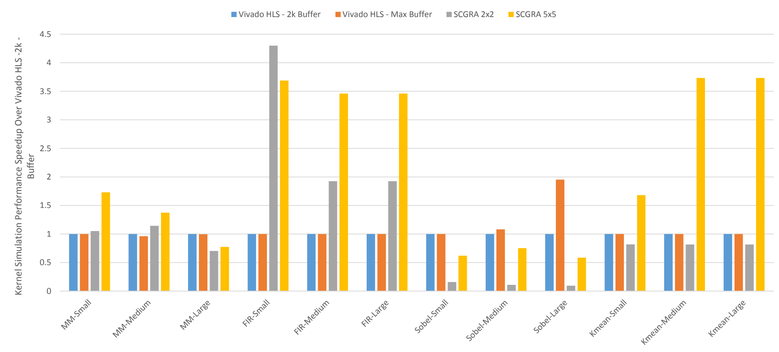

Figure 7: Normalized Kernel Simulation Performance

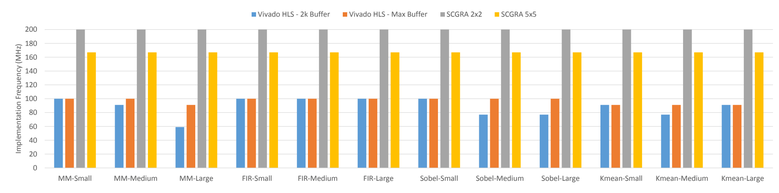

Figure 8: Implementation Frequency

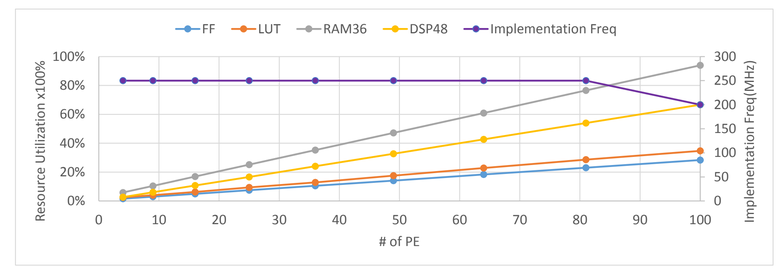

Figure 9: SCGRA Implementation Scalability

Experiment setup

We take four applications including matrix multiplication (MM), FIR filter, Kmean and Sobel edge detector as our benchmark. Each application is further provided with three different data sets as illustrated in Table 1. The benchmark was implemented on Zedboard using both QuickDough and Vivado HLS based design methodology. Since we haven't complete the SCGRA customization work, we just have two typical SCGRA configurations as an example. The configuration details are listed in Table 2.

Benchmark	MM	FIR	Sobel	Kmean
Parameters	Matrix Size	# of Input # of Taps+1	# of Vertical Pixels # of Horizontal Pixels	# of Nodes # of Centroids Dimension Size
Small	10	40/50	8/8	20/4/2
Medium	100	10000/50	128/128	5000/4/2
Large	1000	100000/50	1024/1024	50000/4/2

Table 1: Detailed Configuration of The Benchmark

Topology	SCGRA Size	Data Mem	Inst Mem	In/Out Buffer	Addr Buffer
Torus	2×2, 5×5	256×32bits	1024×72bits	2048×32bits	4096×32bits

Table 2: SCGRA Configuration