CS294-141 Proposal: Communicating Sequential Processes Support for High-Level Synthesis Tools

Qijing Huang, Tan Nguyen, Arya Reais-Parsi
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{qijing.huang, tan.nqd, aryap}@berkeley.edu

I. Overview and Motivation

With the emergence of FPGA Acceleration-as-a-Service offerings from major cloud service providers in the past year Amazon [1], Alibaba [2], Baidu [3], Tencent [4], FPGAs have become more accessible to general application developers. However, ease of use remains a critical barrier to the adoption of FPGAs. High Level Synthesis (HLS) is one potential solution, providing a more familiar environment for software developers and software minded hardware engineers to develop hardware accelerators. HLS allows developers to code an accelerator in a familiar language, such as C, and to compile that program description into RTL. Support of sequential C is maturing in existing HLS frameworks Vivado HLS [5], Altera OpenCL [6], LegUp [7], but the high-level abstraction to support concurrent models of computation is still limited in these tools. Given the complexity of modern applications and the underlying hardware platforms, an expressive yet simple abstraction for concurrency is needed.

In this work, we propose to map a concurrency similar to Communicating Sequential Processes (CSP) to hardware through HLS, and to potentially design a DSL or select a subset of features from existing CSP-inspired languages to map to our intermediate translation layer. CSP is an established formalism for concurrency in computer programs, having been an active area of research since its introduction by Hoare in 1977 [8]. It is similar in many respects to slightly older Actor model. Traditionally, concurrently executing programs must negotiate access to shared memory in the presence of arbitrary interleaving, usually requiring careful management of locks around critical sections. Under CSP and the Actor model, processes within programs execute as if independently, concerned only with their local state. (In CSP these processes are anonymous, while under the Actor model they are themselves named actors.) To communicate with each other or share information CSP usest synchronised unbuffered channels; under the Actor model the channel is implicit (Actors send messages to each other by name) and unbuffered. Under CSP, writes to a channel cannot complete until another process is ready to read the value, whereas Actors do not need to synchronous message writes to anothers read.

Modern programming languages like Go and Rust embed ideas from CSP and Actor systems [9], [10], as higher-level interfaces to concurrency. Using channels as a first-class concurrency primitive avoids the difficulty of (and the human errors introduced by) managing lower-level primitives like mutexes, conditions variables, and so on. These channels can be buffered or unbuffered, with multiple producers and consumers. Crucially, this general idea seems to map neatly to hardware logic implementation. Independent actors/processes can be implemented as independent hardware modules, and the channels interconnecting them as fixed FIFO channels. Asking programmers to write in a CSP/Actor model might make it easier to synthesise hardware acceleration for certain processes, as they themselves have are forced to demarcate the processes states and boundaries more cleanly than under lower-level concurrency models.

II. Related Work

Xilinx Vivado HLS supports channel communication but no concurrency model is enforced. OpenCL supports limited parallelism in a Single Program Multiple Data scheme. Among other tools, LegUp supports the Pthreads/OpenMP model [11] to generate multi-core hardware accelerators on both FPGA-SoC and FPGA-only system. While Pthreads can be used to construct various concurrency models, synchronization through the low-level shared-memory mutex abstraction it provides is problematic. The dependencies of parallel tasks in the program are manifested by accesses to shared variables, which makes it hard to write, understand and debug the programs. Besides, runtime crashes in critical sections will contaminate memory, making it hard to recover from failures. Ultimately, Pthreads/OpenMP support in LegUp assumes the existence of a shared address space, which limits its usage to single machine with shared memory. [12] proposes using Message Passing Interface as a programming model for configurable computers, which could be used to complement the Pthreads/OpenMP model. CSP-like models, on the other hand, could be used to unify the concurrency model for single machines and distributed systems with stricter semantics for inter-process communication and synchronization than in MPI.

III. Objectives

Our high level objectives are as follows:

1) Support a CSP-like concurrency model written in C under high level synthesis.
   a) Create abstractions/libraries for C that enable CSP-like concurrency in a way that is fully synthesisable by either Vivado HLS tools or the LegUp framework (we can build on existing Open Source libraries like [13]
   b) Determine hardware model for communication channel
   c) Implement a small set of applications using our framework, e.g. a binary neural network, md5 computation, Black-Scholes option price simulation, Mandelbrot

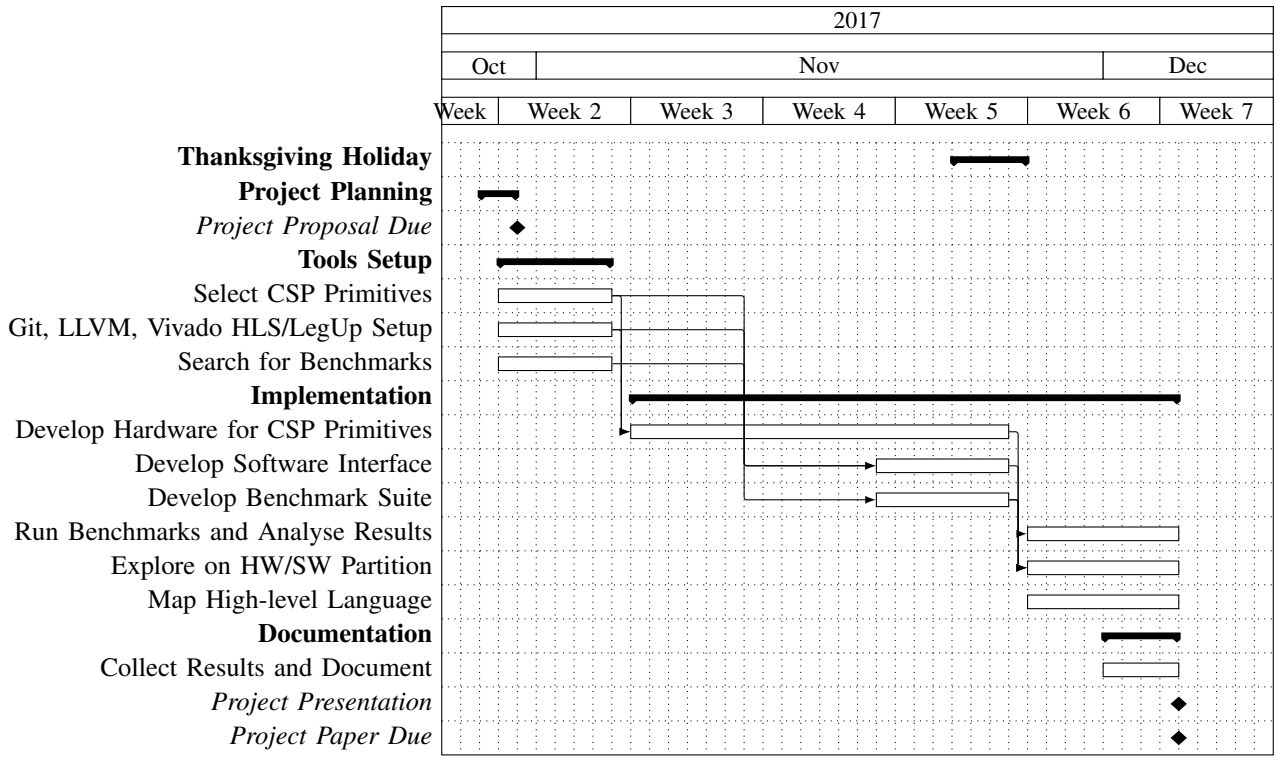

Fig. 1. Proposed Project Schedule

d) Evaluate our programs against the same applications written for pthread-enabled LegUp HLS (in terms of FPGA resource use and application performance), but also against the applications using software concurrency running on the CPU
e) Investigate and evaluate the partitioning of processes in the CSP model across hardware blocks and tasks remaining in software on the CPU; investigate the tradeoffs for extending the CSP-like channel across this interface

2) (Stretch goal) Investigate higher-level expressivity of the concurrency model for synthesis:
   a) Map high-level languages supporting CSP-like constructs to the C form derived in (1), for example Go or Rust; or
   b) Design a limited, embedded DSL for expressing CSP-like concurrency

We hope to show that CSP-like concurrency models are amenable to hardware synthesis and that they perform acceptably well compared to synthesis of software using lower-level synchronisation in some number of dimensions, for example FPGA resource usage, ease of programmability, and application execution time.

Initially, all three team members will investigate and write the abstractions and libraries for CSP-like models in C. Then we will break up to perform each of the sub-objectives (1) (a), (b), (c), (e) independently. For the evaluation task, each piece of software will be pursued by one person. We will re-evaluate the work distribution once (if) we reach the stretch objective (2).

IV. Proposed Schedule

The proposed schedule for the project can be found in Figure 1.

References

[1] "Amazon EC2 F1 Instances." [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[2] "Intel Collaborates with Alibaba Cloud to Help Customers Accelerate Business Applications." [Online]. Available: https://newsroom.intel.com/news/alibaba-collaborating-intel-fpga-based-solution-help-customers-accelerate-business-applications/

[3] "Baidu Deploys Xilinx FPGAs in New Public Cloud Acceleration Services." [Online]. Available: https://www.xilinx.com/news/press/2017/baidu-deploys-xilinx-fpgas-in-new-public-cloud-acceleration-services.html

[4] "Tencent FPGA Cloud." [Online]. Available: https://cloud.tencent.com/product/fpga?lang=en

[5] "Vivado HLS." [Online]. Available: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[6] "Intel OpenCL SDK." [Online]. Available: https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html

[7] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, and J. H. Anderson, "LegUp: An Open-source High-level Synthesis Tool for FPGA-based Processor/Accelerator Systems," *ACM Trans. Embed. Comput. Syst.* [Online]. Available: http://doi.acm.org/10.1145/2514740

[8] C. Hoare, "Communicating sequential processes," *Communications of the ACM*, Aug. 1978.

[9] "The Go Programming Language ," 2017. [Online]. Available: https://golang.org/doc/

[10] "The Rust Programming Language," 2017. [Online]. Available: https://www.rust-lang.org/en-US/documentation.html

[11] J. A. Jongsok Choi, Stephen Brown, "From pthreads to multicore hardware systems in legup high-level synthesis for fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Oct. 2017.

[12] M. Saldaña, A. Patel, C. Madill, D. Nunes, D. Wang, P. Chow, R. Wittig, H. Styles, and A. Putnam, "MPI As a Programming Model for High-Performance Reconfigurable Computers," *ACM Trans. Reconfigurable Technol. Syst.* [Online]. Available: http://doi.acm.org/10.1145/1862648.1862652

[13] "Pure c implementation of go channels." [Online]. Available: https://github.com/tylertreat/chan