



AFRL-RI-RS-TR-2020-117

SPIRAL: FUTURE-PROOF PERFORMANCE PORTABILITY

CARNEGIE MELLON UNIVERSITY

JULY 2020

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2020-117 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

STEVEN DRAGER
Work Unit Manager

/ S /

GREGORY HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE**Form Approved
OMB No. 0704-0188**

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) JULY 2020			2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) NOV 2015 - NOV 2019	
4. TITLE AND SUBTITLE SPIRAL: FUTURE-PROOF PERFORMANCE PORTABILITY					5a. CONTRACT NUMBER FA8750-16-2-0033	
					5b. GRANT NUMBER N/A	
					5c. PROGRAM ELEMENT NUMBER 61101E	
					5d. PROJECT NUMBER BRAS	
6. AUTHOR(S) Franz Franchetti, James Hoe, Jose Moura, David Padua, and Mike Franusich					5e. TASK NUMBER CM	
					5f. WORK UNIT NUMBER UF	
					8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie Mellon University 5000 Forbes Ave Pittsburgh PA 15213					10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505					11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2020-117	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT The SPIRAL/BRASS effort demonstrated that it is possible to develop adaptive-future compatible software that is truly "write once-run everywhere," leading to highest performance at the level of human hand-optimized software for advanced platforms and novel platforms that were not yet known at software development and deployment time. Over the course of this effort, demonstrations for defense-relevant computing platforms and algorithms, which seamlessly adjusted computing power and imaging resolution, while migrating the execution across varying platforms, were performed. The underlying SPIRAL technology is freely available and maintained as open source software.						
15. SUBJECT TERMS SPIRAL, high-performance, portability, compositional formal methods, dynamic transformation of software systems, linguistic abstraction						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			STEVEN DRAGER	
U	U	U	UU	25	19b. TELEPHONE NUMBER (Include area code) N/A	

TABLE OF CONTENTS

LIST OF FIGURES	ii
1.0 SUMMARY	1
2.0 INTRODUCTION	2
2.1 Comparison with Current Technology	3
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	4
3.1 The SPIRAL/BRASS System	4
3.2 SPIRAL	4
4.0 RESULTS AND DISCUSSION	7
4.1 Deliverables Description	7
4.2 Demonstration: Dynamic Range of Target Platforms	8
4.3 Demonstration: Dynamic Reconfiguration of Synthetic Aperture Radar	9
4.4 Demonstration: Flight Test “Bump In The Wire”	10
4.5 Open Source Software: SPIRAL BRASS Package	11
4.6 Technology Transition and Technology Transfer Targets	11
4.7 Dissemination: Presentation	12
4.8 Dissemination: Awards	13
4.9 Dissemination: Publications	13
4.9.1 Journal Papers	13
4.9.2 Conference Papers (Fully Reviewed)	14
5.0 CONCLUSIONS	16
LIST OF ACRONYMS AND ABBREVIATIONS	17

LIST OF FIGURES

Figure 1. Components of the SPIRAL/BRASS System.....	4
Figure 2. SPIRAL/BRASS Dynamic Range.....	8
Figure 3. Reconfigurable SAR Kernel.....	9
Figure 4. Notional Post-Deployment and In-Mission Adaptations	10
Figure 5. Open Source SPIRAL.....	11

1.0 SUMMARY

This is the final report of the Defense Advanced Research Projects Agency (DARPA) Building Resource Adaptive Software Systems (BRASS) SPIRAL/BRASS effort at Carnegie Mellon University with team members from the University of Illinois Urbana-Champaign and SpiralGen, Inc. The project had three phases with end-of-phase demonstrations. The project leave-behind includes the final demonstration and open source software. The key demonstration of our effort was that it is possible for the same unmodified source code to run across a wide range of platforms with vastly varying architectural, performance, power, and parallelism characteristics. The source language in this technology demonstration is C++ with a properly chosen mathematical library infrastructure. The target platforms span a wide range of processors and accelerators with a dynamic range of five to six orders of magnitude and a release date spread over 20 years. The project tag line is “*one source code, one tool, always highest performance.*”

Over the course of the project we demonstrated this capability for various notional defense-relevant computing platforms and algorithms, by seamlessly adjusting computing power and imaging resolution while migrating the execution across varying platforms. As end-of-project demonstration the technology was wrapped up as a small appliance for flight that was called a “*bump in the wire.*” The appliance filters on-platform network traffic during a flight and pre-analyzes raw data before transmission to the base station. This is done to accommodate base station hardware, emulating overcoming hardware not capable of fully processing the high- resolution on-board data streams. The behavior of the appliance is controlled from the ground via configuration messages that trigger our system to perform the necessary reconfiguration. The demonstrations are available via the system integrator South West Research Institute (SwRI). The underlying SPIRAL system and the BRASS package is available as open source software with permissive Berkeley Software Distribution (BSD) style license from the SPIRAL web page, maintained by SpiralGen, Inc.

2.0 INTRODUCTION

Software implementations and algorithms that solve a particular problem are inherently not *future-proof* as an algorithm appropriate for the problem *and* execution environment has to be instantiated into software that locks in many choices. However, the actual *operation* – the answer for a given input – remains invariant through execution environment changes and changes in required fidelity. We developed a layered approach to application specification which leverages this insight: we capture the systematic process by which the specification of a desired operation is turned into an algorithm, and subsequently instantiated into an actual implementation. We targeted a class of problems that is captured through regular computation over multi-dimensional data. Important signal and image processing operations that run on *high performance embedded computing* (HPEC) platforms. In this class are also simulations and computer-aided design/engineering (CAD/CAE) problems on traditional high performance computing (HPC) platforms and embedded application kernels. For these problems, we developed a layered approach to specifying 1) the desired operation, 2) the *paradigm* (environment) in which the operation is computed, and 3) the specific details of the paradigm i.e. the industry standard architecture (ISA), cache sizes, and the type of multi-threading available. These three layers map to changes required in the operation, algorithms, and implementations respectively. An implication of separating the specification into these three distinct but inter-related layers is that there is a natural separation of concerns, allowing the same specification to handle different changes to the environment. For example, when porting the operation to an architecture with a different ISA, this only affects the regions of the code touched by the platform specification, whereas changing from a sequential to multicore environment requires changes to the choice of algorithm.

Our approach centers on the ideas that future-proof code requires a *layered specification* of semantics, quality/parameter relationships, performance/quality/resource trade-offs, quality measures and fault tolerance, and a powerful code synthesis engine that turns this specification into reconfigurable, extensible high performance implementations specialized to their execution environment. We extend the SPIRAL code generation and autotuning system and philosophy, augmented with lessons learned from the Fastest Fourier Transform in the West (FFTW), Belmann Logistics Information (BLIS), Aggregate Table Language and System Computing (ATLAS), Hierarchically Tiled Arrays (HTA), PetaBricks, and modern as well as traditional optimizing compiler approaches. The key contribution is to turn SPIRAL's extreme cross-platform compile-time portability into an extensible post-deployment reconfiguration capability. SPIRAL becomes akin to an installation/reconfiguration-time Just-in Time (JIT) compiler that regenerates an FFTW-like implementation for a given specification and execution target. However, the implementation is not used as an autotuning library but as a framework to reconfigure the application based on execution environment changes and changes in user/mission requirements. Finally, we

define a user-friendly specification language that borrows from HTA, and functional programming ideas to allow users to fully express application intent and trade-offs as well as execution environments. To provide an even easier on-ramp for legacy code, a subset of the functionality will be accessible through a delayed-evaluation interface using Basic Linear Algebra subprograms (BLAS), Linear Algebra Package (LAPACK), FFTW and a subset of C as domain-specific specification language.

2.1 Comparison with Current Technology

Many attempts (e.g. FFTW, ATLAS, Optimized Sparse Kernel Interface (OSKI), hierarchically tiled arrays, PetaBricks, Ccitt high level language (CHiLL) and Procedure Language for Users in Test and Operations (PLUTO)/PrimeTile) at performance portability through the use of autotuning/program generation or compiler-level transformations suffer from the limitation that they cannot port to or require substantial effort to port to novel, not-yet-invented future architectures and machines. Assumptions are baked into most autotuning frameworks to make the search space tractable, which may result in suboptimal results (if any) as solutions for future platforms may be eliminated from the search space. Traditional compilers suffer from the fact that they work with code, not operation specifications, resulting in the obfuscation of application intent. Annotation-aware compilers require manually provided annotations, which do not provide correctness guarantees. Library-based approaches (e.g. FFTW and BLAS) can be seen as domain-specific languages (DSL) but are restricted in scope, whereas our proposed class of operations often requires cross-domain interactions.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 The SPIRAL/BRASS System

The main components of the SPIRAL/BRASS system are depicted in Figure 1. In a user C/C++ program, library hooks abstract the system and allow the user to interact with the reconfiguration system in a transparent and well-understood way. During execution, library calls are trapped by the SPIRAL module, and in a runtime/initialization time, JIT code generation pass builds adapted high performance implementations of the computation specified through a sequence of library calls. The code generation process is parameterized by plug-ins that capture hardware aspects and can be installed post-deployment. This approach allows the system to be “future compatible”, that is, the specified computation can be mapped efficiently to hardware that was not known at program compilation and deployment time.

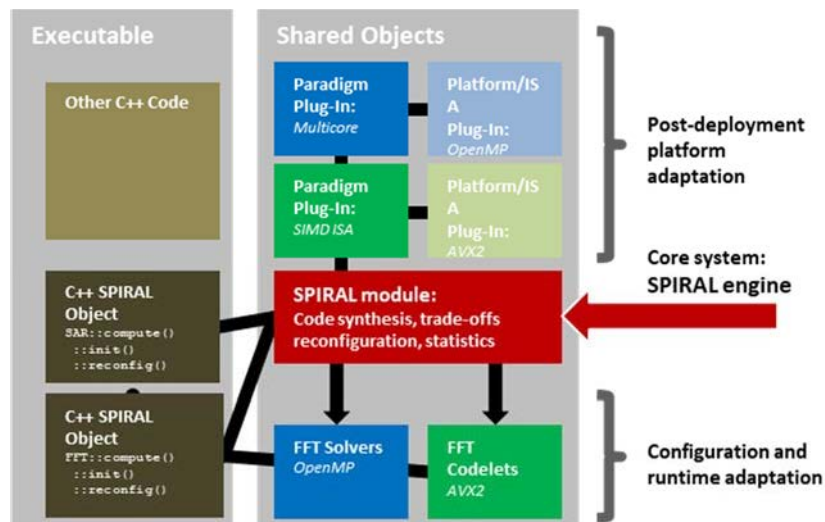


Figure 1. Components of the SPIRAL/BRASS System

3.2 SPIRAL

SPIRAL builds on 20 years of research by principal investigators Franchetti, Hoe, and Low, and 9 years of commercial research and development (R&D) in the SPIRAL effort. The SPIRAL system has initially been open sourced as part of the DARPA Power Efficient Revolution for Embedded Computing Technologies (PERFECT) program, and is now available under a permissive BSD license on GitHub. The process of open sourcing is ongoing as documentation and new packages that have been developed in the past and are now made available are continuously added. The SPIRAL/BRASS package is one of these maintained SPIRAL packages.

3.2.1 Application domains. SPIRAL has demonstrated that it can capture a wide range of algorithms including communication algorithms like the full physical layer of wireless fidelity (WiFi) and global system for mobile communication (GSM)/long term evolution (LTE), control algorithms, linear algebra kernels like matrix multiplication and line unit (LU) factorization, statistical algorithms, ordinary differential equation (ODE) integration and approximation, partial differential equation (PDE) solver kernels, and a range of graph/artificial intelligence (AI)/machine learning (ML) algorithms. This range of application domains far exceeds SPIRAL's initial focus for the domain of linear signal processing transforms (the fast Fourier transform and its variants), and higher level image formation algorithms like synthetic aperture radar (SAR) and space-time adaptive processing (STAP).

3.2.2 Performance portability. SPIRAL has demonstrated performance portability across a wide range of hardware architectures and scales. The same SPIRAL program specifying a computational kernel can be mapped from small scale (Raspberry Pi or cell phone) to HPEC scale (single fat node to single rack of multicore central processing units (CPUs) and graphical processing units (GPUs)) all the way up to HPC/supercomputing scale (100,000+ cores), and the produced implementations outperforms the best human programmers. SPIRAL has successfully targeted a wide range of single core/multicore/many core CPUs, GPUs, digital signal processors (DSPs), the Cell BE, Xeon PHI, field programmable gate arrays (FPGAs), and in pre-silicon settings (IBM Cell BE and Blue Gene Q (BG/Q), and Intel advanced vector extensions (AVX) and Xeon PHI). This demonstrates SPIRAL's performance portability capabilities based on automatic performance tuning as well as on machine models.

3.2.3 Scalability. SPIRAL has shown scalability up to a large core count on a range of clusters and supercomputers: high performance code was generated for standard HPC and commodity clusters, BlueGene/L/P/Q (up to 128k cores), the Japanese K computer (88k CPUs), and Blue Waters (50k cores, 4k GPUs). SPIRAL is currently used in the Department of Energy (DOE) Exascale Computing Project to target Summit (200 Pflop/s, 200k cores, 27k GPUs), and is slated for targeting the upcoming Exascale machines (Aurora and El Capitan) as soon as they become accessible to us through our DOE collaborators. In 2010 SPIRAL won the DARPA High Productivity Computing Systems (HPCS) Class II Challenge for Global Fast Fourier Transform (FFT) on the BG/Q supercomputer. SPIRAL was used to develop the FFT library for BlueGene/L that was part of the QBox system that won the Gordon Bell Prize in 2006.

3.2.4 Peak performance. SPIRAL was designed to automatically deliver the performance of the best hand-tuned code on a new target platform on day one. This was shown numerous times over the last 2 decades in comparisons to highly-tuned vendor libraries across a wide range of platforms (Intel Math Kernel Library (MKL) and Integrated Performance Primitives (IPP), Advanced Micro Devices (AMD) Core Math Library (ACML), International Business Machines (IBM) Engineering and Scientific Subroutine Library (ESSL), Mercury scientific algorithm library (SAL), among others) and open source libraries (FFTW, Karn's Viterbi decode library, and others). For well-studied kernels like 2-power FFTs, SPIRAL produces

code as good as the best human hand-tuned code. For less common library kernels or kernels that are hard to map to advanced instruction sets, SPIRAL may produce code that is 5x faster than the best available expert hand tuned high performance library code.

3.2.5 Backends. To provide performance portability, SPIRAL has targeted a wide range of execution environments that implement various programming models. Among others, SPIRAL has targeted language extensions (Open Multi-Processing (OpenMP), Open Accelerators (OpenACC), Open Computing Language (OpenCL), Compute Unified Device Architecture (CUDA)), communication and threading libraries and schedulers (Portable Operating System Interface threads (pthreads), Cilk, Message Passing Interface (MPI)), low-level instruction sets like Streaming single instruction multiple data (SIMD) Extensions (SSE)/AVX/AVX512/NEON/QPX/VMX/SVX and their variants (via intrinsic functions and inline assembly). SPIRAL supports automatic performance tuning, and hardware profile-guided optimization and has been used as backend codelet generator for the polyhedral system Pluto/PrimeTile.

4.0 RESULTS AND DISCUSSION

4.1 Deliverables Description

We demonstrated the following SPIRAL/BRASS prototype components as an open source package of the SPIRAL system, available at www.spiral.net: 1) A DSL embedded in C++ that captures— for the targeted application domain—application intent, application semantics, quality/parameter dependencies, and the quality/resource/performance trade-off space. 2) A DSL capturing structural and quantitative hardware properties and execution environments. 3) An internal representation that unifies these DSLs into one framework enabling application code synthesis and traversal of the trade-off space as required by the execution environment and mission requirements. 4) A domain specific code synthesis tool “configuration time JIT” that synthesizes from the internal representation a C++ object implementation and achieves performance similar to hand-optimized math libraries across a wide range of platforms while providing the BRASS-required reconfigurability. 5) A plug-in mechanism to update the deployed executable (containing the code synthesis JIT) to support new platforms, paradigms, and ISAs. 6) Prototypical applications (polar formatting SAR and bump-in-the-wire adaptive processing) built for Intel multicores, and re-deployed on a different CPU architecture accelerator (Advanced Reduced Instruction Set Computing (RISC) Machine (ARM)). Example code has also been deployed on an FPGA.

4.2 Demonstration: Dynamic Range of Target Platforms

As part of the BRASS program we demonstrated a wide range of adaptability *from the same source code*. As shown in Figure 2 below, SPIRAL/BRASS is able to adapt and target from one source code with one tool platforms that were introduced across 20 years, from 1W to 10 kW power envelope, and with computational power of less than 1 Gflop/s up to 200 TFlop/s. The targeted platforms span the range from CPUs to GPUs and FPGAs from small-scale embedded devices to large server-class multicore CPUs and accelerators.

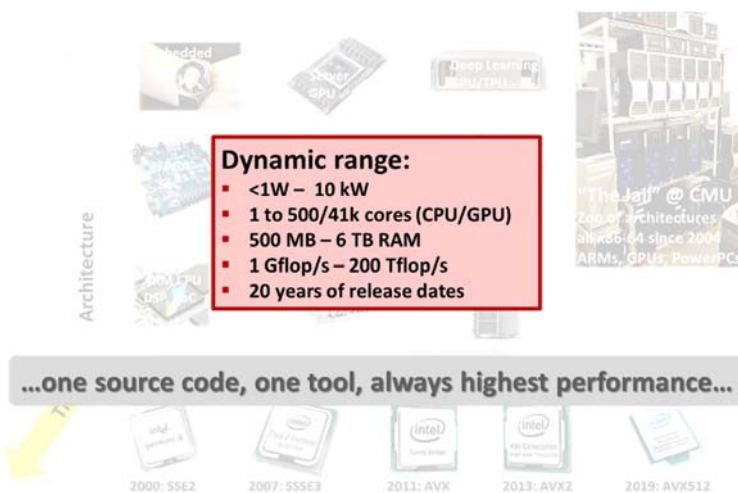


Figure 2. SPIRAL/BRASS Dynamic Range

4.3 Demonstration: Dynamic Reconfiguration of Synthetic Aperture Radar

In the BRASS program we demonstrated dynamic reconfiguration of polar formatting SAR without interrupting streaming processing. As depicted in Figure 3, we demonstrated a reconfigurable SAR kernel that could be switched to various operating points (resolutions, resampling factors, hardware resources, target platform). We demonstrated that post-deployment new hardware components could be installed (starting from Intel multicore CPUs we added FPGAs and ARM boards), and the software would switch over to the newly installed components without losing a frame. After installation, the SAR kernel was in the background re-optimized for the new hardware and when the code generation and optimization was completed the kernel was activated and computation switched over seamlessly, all without recompiling the user SAR program.

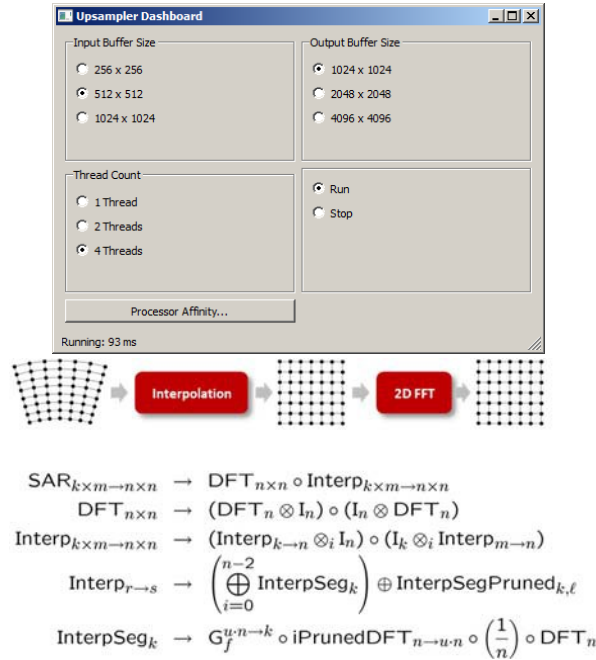


Figure 3. Reconfigurable SAR Kernel

4.4 Demonstration: Flight Test “Bump In The Wire”

The major end-of-program demonstration showed a fully deployable prototype developed between Carnegie Mellon University, SpiralGen, Inc., and SwRI. We developed and demonstrated a “bump in the wire” configurable computational filter for the Flight Test Challenge Problem. A Raspberry PI ARM board with local area network (LAN) adapter is inserted between onboard sensors and the radio downlink in a flight test. The ARM board acts as self-contained intelligent filter. High bandwidth data sent by the sensor is processed and down-sampled so that legacy ground stations can process and display the data. The exact filtering behavior is defined by a Microsoft Data Link (MDL) configuration file that is sent to the device. Upon receipt the SPIRAL/BRASS system interprets the MDL file and generates a new filter kernel according to the specification. Once activated the filter seamlessly performs the specified operation, enabling the legacy system to extract maximum knowledge from the flight test. This reconfiguration is enabling advanced filtering and in-air pre-processing, as notionally shown in Figure 4.

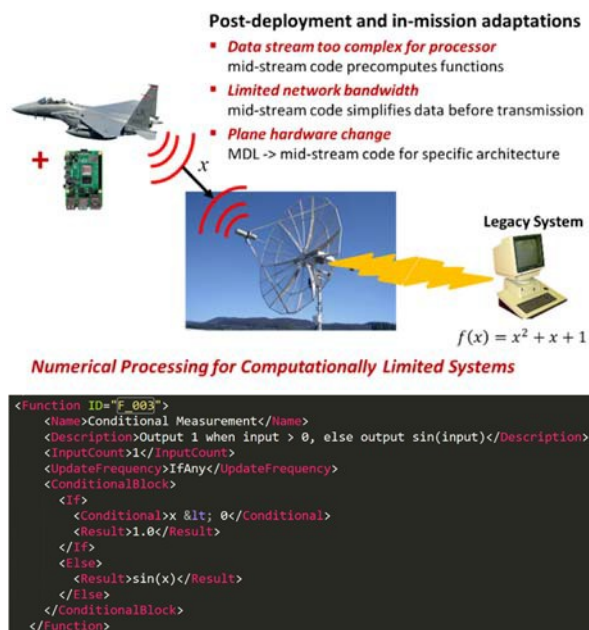
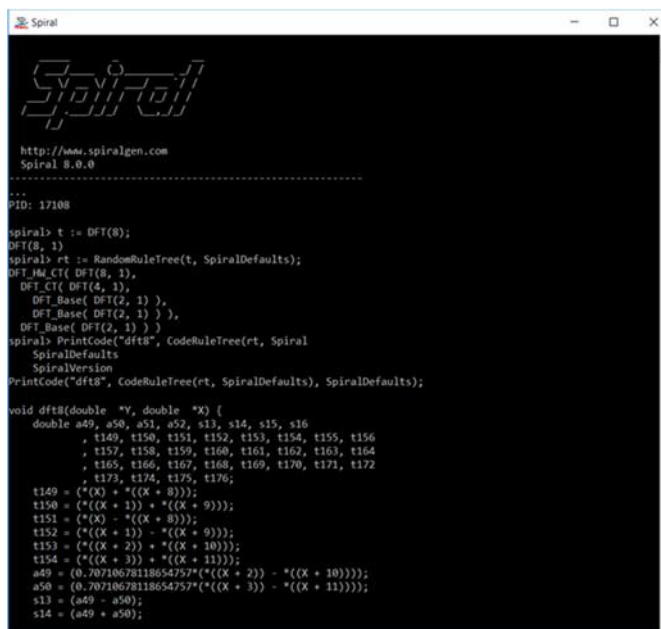


Figure 4. Notional Post-Deployment and In-Mission Adaptations

4.5 Open Source Software: SPIRAL BRASS Package

The SPIRAL system (<http://www.spiral.net>) has been open-sourced under a permissive BSD license and all SPIRAL components developed under BRASS are available under BSD Open Source as GitHub-hosted SPIRAL package maintained by SpiralGen, Inc.



```
Spiral
-----
http://www.spiralgen.com
Spiral 8.0.0
-----
PID: 17108

spiral> t := DFT(8);
DFT(8, 1)
spiral> rt := RandomRuleTree(t, SpiralDefaults);
DFT_IM_CT( DFT(8, 1),
  DFT_CT( DFT(4, 1),
    DFT_Base( DFT(2, 1) ),
    DFT_Base( DFT(2, 1) ) ) )
spiral> PrintCode("dft8", CodeRuleTree(rt, Spiral
SpiralDefaults
SpiralVersion
PrintCode("dft8", CodeRuleTree(rt, SpiralDefaults), SpiralDefaults);

void dft8(double *Y, double *X) {
  double a49, a50, a51, a52, s13, s14, s15, s16
    + t149, t150, t151, t152, t153, t154, t155, t156
    + t157, t158, t159, t160, t161, t162, t163, t164
    + t165, t166, t167, t168, t169, t170, t171, t172
    + t173, t174, t175, t176;
  t149 = *(X + *(X + 8));
  t150 = (*(X + 2)) + *(X + 9));
  t151 = *(X - *(X + 8));
  t152 = *(X + 2) - *(X + 9));
  t153 = (*(X + 2)) + *(X + 10));
  t154 = *(X + 3) + *(X + 11));
  a49 = (0.70710678118654757*(*(X + 2) - *(X + 10)));
  a50 = (0.70710678118654757*(*(X + 3) - *(X + 11)));
  s13 = (a49 - a50);
  s14 = (a49 + a50);
```

Figure 5: Open Source SPIRAL

4.6 Technology Transition and Technology Transfer Targets

SpiralGen, Inc., founded in 2009, maintains the open source SPIRAL system, available from the SPIRAL web page and from GitHub. SPIRAL functionality is continuously hardened and transferred from university grade graduate student software to hardened industry-strength open source software.

4.7 Dissemination: Presentation

Franchetti, F. “SPIRAL, FFTX, and Automating the Optimization of the Fast Fourier Transform” *Austrian High-Performance Computing Meeting*, Institute of Science and Technology Austria (IST Austria), Klosterneuburg, Austria (2020 Feb 19), ***Invited Opening Keynote***.

Franchetti, F. “FFTX and SpectralPACK,” *SIAM Conference on Computational Science and Engineering*, Seattle, WA (2020 Feb 14).

Franchetti, F. “FFTX and SpectralPACK,” *Supercomputing 2019*, Denver, CO (2019 Nov 18).

Franchetti, F. “SPIRAL: AI for High-Performance Code,” *Oak Ridge National Laboratory*, Oak Ridge, TN (2019 Oct 10).

Franchetti, F. “HPC: Bridging ECE and DBMI,” *Department of Biomedical Informatics, University of Pittsburgh*, Pittsburgh, PA (2019 Oct 4).

Franchetti, F. “FFTX and SpectralPACK,” *Exascale Computing Project CoPA*, Santa Fe, NM (2019 Sep 5).

Franchetti, F. “SPIRAL’s Operator Language: From Textbook Math to High Performance – With Correctness Guarantees,” *University of Nagoya*, Nagoya, Japan (2019 Aug 8).

Franchetti, F. “FFTX and SpectralPACK: A First Look,” *University of Vienna*, Vienna, Austria (2019 Jun 17).

Franchetti, F. “SPIRAL’s Operator Language: From Textbook Math to High Performance – With Correctness Guarantees,” *Vienna University of Technology*, Vienna, Austria (2019 Jun 18).

Franchetti, F. “SPIRAL’s Operator Language: From Textbook Math to High Performance – With Correctness Guarantees,” *International Federation for Information Processing Working Group*, Salem, MA (2019 Apr 29).

Franchetti, F. “FFTX and SpectralPACK: A First Look,” *SIAM Conference on Computational Science and Engineering*, Spokane, WA (2019 Mar 1).

Franchetti, F. “SPIRAL, FFTX, and the Path to SpectralPACK,” *Supercomputing 2018*, Dallas, TX (2018 Nov 13).

Franchetti, F. “Formal Software Synthesis of Computational Kernels,” *Advanced Supercomputing Environment (ASE) Seminar, University of Tokyo, Tokyo, Japan* (2018 Jul 27).

Franchetti, F. “SPIRAL FFT Library Generation and Autotuning,” *SIAM Conference on Parallel Processing for Scientific Computing, Waseda University, Tokyo, Japan* (2018 Mar 8).

Franchetti, F. “Specialized, perhaps configurable, hardware and software are necessary to achieve high-performance, scalable data analytics,” *Seventh Workshop on Irregular Applications: Architectures and Algorithms (IA³), in conjunction with Supercomputing 2017 Denver, CO* (2017 Nov 13) **Invited Panelist**.

Franchetti, F. “High-Performance Computing Libraries as Domain-Specific Language,” *Vienna University of Technology, Vienna, Austria* (2016 July 04), **Invited Seminar**.

Franchetti, F. “Formal Software Synthesis of Computational Kernels,” *Swanson Engineering Seminar, University of Pittsburgh, Pittsburgh, PA* (2016 March 16).

4.8 Dissemination: Awards

Franchetti, F. and Ruzicka, V. **Best Paper Finalist**, *High Performance Extreme Computing (HPEC)*, 2018.

Franchetti, F., Zhang, J., Spampinato, D. G. and McMillan, S., **MIT GraphChallenge Finalist**, *High Performance Extreme Computing (HPEC)*, 2018.

Franchetti, F., Low, T. M., Spampinato, D. G., Kutuluru, A., Sridhar, U., Popovici, D. T., and McMillan, S., **MIT GraphChallenge Finalist**, *High Performance Extreme Computing (HPEC)* 2018.

4.9 Dissemination: Publications

4.9.1 Journal Papers.

Bolten, M., Franchetti, F., Kelly, P., Lengauer, C., Mohr, M., “Algebraic Description and Automatic Generation of Multigrid Methods in SPIRAL,” *Concurrency and Computation: Practice and Experience*, 2017.

Franchetti, F., Moura, J. M. F., Padua, D. A., Dongarra, J. “Scanning the Issue: From High-Level Specification to High-Performance Code,” *Proceedings of the IEEE*, **106**, 11, 2018. Special Issue on *From High Level Specification to High Performance Code*.

Franchetti, F., Low, T. M., Popovici, D. T., Veras, R. M., Spampinato, D. G., Johnson, J. R., Püschel, M., Hoe, J. C. Moura, J. M. F., “SPIRAL: Extreme Performance Portability,” *Proceedings of the IEEE*, **106**, 11, 2018. Special Issue on *From High Level Specification to High Performance Code*.

4.9.2 Conference Papers (Fully Reviewed)

Franchetti, F., Spampinato, D. G., Kulkarni, A., Popovici, D. T., Low, T. M., Franusich, M., Canning, A., McCorquodale, P., Van Straalen, B., Colella, P., “FFTX and SpectralPack: A First Look,” *IEEE International Conference on High Performance Computing, Data, and Analytics*, Bengaluru, India (2018).

Koops, H. V., Garg, K., Kim, M., Li, J., Volk, A., Franchetti, F., “Multirotor UAV State Prediction Through Multi-microphone Side-Channel Fusion,” *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Daegu, Korea (2017).

Low, T. M., Rao, V. N., Lee, M., Popovici, D., Franchetti, F., McMillan, S., “First Look: Linear Algebra-based Triangle Counting without Matrix Multiplication,” *IEEE High Performance Extreme Computing*, Waltham, MA (2017).

Low, T. M., Spampinato, D. G., Kutuluru, A., Sridhar, U., Popovici, D. T., Franchetti, F., McMillan, S., “Linear Algebraic Formulation of Edge-Centric K-truss Algorithms with Adjacency Matrices,” *IEEE High Performance Extreme Computing Conference*, Boston, MA (2018), **MIT GraphChallenge Finalist**.

Popovici, D. T., Franchetti, F., Low, T. M., “Mixed Data Layout Kernels for Vectorized Complex Arithmetic,” *IEEE High Performance Extreme Computing Conference*, Waltham, MA (2017).

Popovici, D. T., Low, T. M., Franchetti, F., “Large Bandwidth-Efficient FFTs on Multicore and Multi-Socket Systems,” *IEEE International Parallel & Distributed Processing Symposium*, Vancouver, Canada (2018).

Ruzicka, V., Franchetti, F., “Fast and Accurate Object Detection in High Resolution 4K and 8K Video Using GPUs,” *IEEE High Performance Extreme Computing Conference*, Boston, MA (2018), **Best Paper Finalist**.

Takahashi, D., Franchetti, F., “FFTE on SVE: SPIRAL-Generated Kernels,” *International Conference on High Performance Computing in Asia-Pacific Region*, Fukuoka, Japan (2020) pp. 114-122.

Veras, R. M., Franchetti, F., “A Scale-Free Structure for Real World Networks,” *IEEE High Performance Extreme Computing Conference*, Waltham, MA (2017).

Zaliva, V., Franchetti, F., “HELIX: A Case Study of a Formal Verification of High Performance Program Generation,” *Workshop on Functional High Performance Computing*, St. Louis, MO (2018).

Zhang, J., Franchetti, F., Low, T. M., “High Performance Zero-Memory Overhead Direct Convolutions,” *International Conference on Machine Learning*, Stockholm, Sweden (2018).

Zhang, J., Lu, Y., Spampinato, D. G., Franchetti, F., “FESIA: A Fast and Efficient Set Intersection Approach on Modern CPUs,” *IEEE International Conference on Data Engineering* Dallas, TX (2020).

Zhang, J., Spampinato, D. G., McMillan, S., Franchetti, F., “Preliminary Exploration on Large- Scale Triangle Counting in Shared-Memory Multicore System,” *IEEE High Performance Extreme Computing Conference*, Boston, MA (2018), **MIT GraphChallenge Finalist**.

5.0 CONCLUSIONS

The SPIRAL/BRASS effort led by Carnegie Mellon University successfully demonstrated goals of the DARPA Building Resource Adaptive Software Systems program. BRASS set out to develop “software that survives 100 years”. In that context, our effort aimed to prove that it is possible to develop adaptive-future compatible software that is truly “write once-run everywhere.” Our efforts demonstrated success, with highest performance, on the level of human hand-optimized software, for advanced platforms and novel platforms that were not yet known at software development and deployment time. Moreover, we demonstrated this capability for defense-relevant software. The underlying SPIRAL technology is freely available as maintained open source software and already serves as the underpinnings of a number of advanced research and development efforts. As a successful project, the main points have been established and the developed technology is being transitioned and disseminated.

LIST OF ACRONYMS AND ABBREVIATIONS

ACML: AMD Core Math Library

AI: artificial intelligence

AMD: Advanced Micro Devices

ARM: Advanced RISC Machine. RISC stands for reduced instruction set computing

ATLAS: Aggregate Table Language and System Computing

AVX: Advanced Vector Extensions (by Intel)

BG/Q: Blue Gene Q

BLAS: Basic linear algebra subprograms

BLIS: Beltmann Logistics Information System

BRASS: Building Resource Adaptive Software Systems

BSD: Berkeley software distribution

C: stands for the letter “C” or “combined”

CAD: computer-aided design

CAE: computer-aided engineering

CHiLL: Ccitt high level language

COPA: Co-Design Center for Particle Applications

CPU: central processing unit

CUDA: Compute Unified Device Architecture (by Nvidia)

DARPA: Defense Advanced Research Projects Agency

DBMI: Department of Biomedical Informatics

DoE: Department of Energy

DSL: domain specific language

DSP: digital signal processor

ECE: Electrical and Computer Engineering

FFT: fast Fourier transform

FFTW: “Fastest Fourier Transform in the West”

FLOPS: floating point operations per second

FPGA: field programmable gate array

Gflops: gigaflops

GitHub: [not an acronym] a web-based hosting service for software development projects that use the *Git* revision control system

GPU: graphics processing unit

GSM: global system for mobile communication

HPC: high performance computing

HPCS: High Productivity Computing Systems

HPEC: high performance embedded computing

HTA: hierarchically tiled arrays

HTA: hyper-text application

IBM: International Business Machines

IBM ESSL: Engineering and Scientific Subroutine Library (by IBM)

IPP: Integrated Performance Primitives (by Intel)

ISA: industry standard architecture

JIT: just-in-time (also called dynamic translation)

k: kilobyte

LAN: local-area network

LAPACK: Linear Algebra Package

LTE: long term evolution

LU: line unit

MDL: Microsoft Data Link

MIT: Massachusetts Institute of Technology

MKL: Math Kernel Library (by Intel)

ML: machine learning

MPI: Message Passing Interface

NEON: [not an acronym] (by Arm)

ODE: ordinary differential equation

OpenACC: Open Accelerators

OpenCL: Open Computing Language

OpenMP: Open Multi-Processing

OSKI: Optimized Sparse Kernel Interface

PDE: partial differential equation

PERFECT: Power Efficient Revolution for Embedded Computing Technologies

PLUTO: Procedure Language for Users in Test and Operations

PrimeTile: a parametric multi-level tiler

Pthreads: Portable Operating System Interface (POSIX) Threads

R&D: research and development

SAL: scientific algorithm library

SAR: synthetic aperture radar

SIAM: Society for Industrial and Applied Mathematics

SIMD: single instruction, multiple data

SPIRAL: [not an acronym] a code generation and autotuning system

SSE: Streaming SIMD Extensions

STAP: space-time adaptive processing

SwRI: Southwest Research Institute

Tflops: teraflops

VMX: VMWare Configuration File

WiFi: wireless fidelity