ELIZABETH PIPKIN (243611)
CHRISTINE PEEK (234573)
JAMES GIACCHETTI (307117)
McMANIS FAULKNER
a Professional Corporation
50 West San Fernando Street, 10th Floor
San Jose, California 95113
Telephone: (408) 279-8700
Facsimile: (408) 279-3244
Email: jgiacchetti@mcmanislaw.com

Attorneys for Plaintiff,
Zeidman Technologies, Inc.

UNITED STATES COURT OF FEDERAL CLAIMS

Zeidman Technologies, Inc.

Plaintiff,

vs.

United States of America,

Defendant.

BID PROTEST

Plaintiff, Zeidman Technologies, Inc. ("Zeidman Technologies"), alleges as follows:

NATURE OF THE CASE

1. Zeidman Technologies is a highly qualified Silicon Valley computing firm, with strong expertise in developing computing tools. Zeidman Technologies develops hardware and software codesign tools for embedded system development that promise to change the way systems are designed, simulated, and implemented. Zeidman Technologies' founder and CEO, Robert Zeidman, is recognized as a pioneer in the fields of software synthesis and analysis. Mr. Zeidman holds 22 patents and has written five textbooks, including Introduction to Verilog, Verilog Designers' Library, and Designing with FPGAs and CPLDs. He holds a Master's degree in electrical engineering from Stanford University and bachelor's degrees in physics and electrical engineering from Cornell University.

2. Despite the fact that Zeidman Technologies owns unique, patented technology that specifically meets the requirements of a Small Business Innovation Research project, Topic AF161-088-1073, the United States Government rejected Zeidman Technologies for Topic AF161-088-1073, entitled "Integrated Code Base and High Performance Embedded Computing Tool." After Zeidman Technologies pressed Defendant to obtain an explanation for the rejection, over the course of months, Defendant provided three (3) conflicting and contradictory explanations, all of which relied on criteria not included in the Small Business Innovation Research program ("SBIR") proposal as a basis for the rejection. Defendant failed to evaluate evidence presented in Zeidman Technologies' proposal.

3. Government contracts must be awarded to the most highly qualified applicants, based upon an evaluation of explicitly stated criteria. Reliance on unstated criteria in the Defendant's procurement decision resulted in an arbitrary decision, and denied Zeidman Technologies a lucrative business opportunity. Zeidman Technologies challenges all procurements for Proposal F161-088-1073.

JURISDICTION AND STANDING

4. Zeidman Technologies brings this Complaint pursuant to 28 U.S.C. § 1491. This Court has jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1491.

5. Zeidman Technologies has standing as an interested party under 28 U.S.C. § 1491. Zeidman Technologies submitted a bid proposal in response to a government solicitation, specifically, Topic AF161-088-1073. Defendant's procurement decision was arbitrary, in that it relied on criteria not stated in the solicitation, and failed to evaluate evidence presented. Defendant offered an explanation for its decision that contradicts the evidence before the agency, and the solicitation submitted by Zeidman Technologies. The criteria Zeidman Technologies allegedly cannot meet was never addressed in the solicitation, amounting to an unstated criterion. But for the alleged procurement errors, Zeidman Technologies would have had a substantial chance to receive the contract.

///

///

PARTIES

6. Zeidman Technologies is a privately held corporation, with its principal place of business at 15565 Swiss Creek Lane, Cupertino, California, USA.

7. Defendant UNITED STATES OF AMERICA is the United States acting through the Air Force.

GENERAL ALLEGATIONS

8. Zeidman Technologies submitted a bid for an SBIR grant with a proposal that met the requirements of the topic. When the proposal was rejected, Zeidman Technologies had to press for months to get a reason. Defendant then supplied three conflicting reasons for the rejection, none of which reflected the stated criteria in the solicitation. In fact, Zeidman Technologies fully met the stated criteria. Although Zeidman Technologies' patented technology does exactly what was described in the proposal, two grants were given to two companies with no known expertise in the required areas. Defendant did not rely on its stated criteria in awarding the contracts.

9. Zeidman Technologies was rejected by Defendant for Proposal F161-088-1073 in response to Topic AF161-088 entitled "Integrated Code Base and High Performance Embedded Computing Tool." Attached hereto as **Exhibit A** is a true and correct copy of the Solicitation.

10. The deadline for submitting a proposal was February 17, 2016, which Zeidman Technologies met. Attached hereto as **Exhibit B** is a true and correct copy of Zeidman Technologies' proposal. Zeidman Technologies received a notification on May 16, 2016 that it did not receive the award. Zeidman Technologies sent a timely letter requesting an explanation for the rejection (i.e., "debriefing"). No one responded with any explanation for a few months. Zeidman Technologies had to twice request a response to its letter.

11. Finally, on August 29, 2016, Zeidman Technologies received an email addressing the rejection. According to the email, Zeidman Technologies' proposal did not meet the criteria of the project.

///

///

12. In reality, Zeidman Technologies met the topic criteria. The SBIR topic description stated: "This topic requests the development of a tool that accepts C-language code for a suite of algorithms and outputs optimized code that can be compiled for a selected device. In addition to making it easy and quick to port algorithms across different platforms for comparing the performance of different architectures, the tool should also allow software developers to concentrate on algorithmic advances rather than processor architecture peculiarities."

13. The August 29, 2016 email states that the project is for converting from software code to a hardware description language (HDL), but that is not true. A hardware description language is not mentioned anywhere in the topic or in the references, which do discuss software. In fact, the topic description would not make sense with respect to HDLs because HDLs are synthesized, not compiled; HDLs create a platform rather than get ported to a platform; and HDLs are used by hardware engineers, not software developers. Robert Zeidman, the President of Zeidman Technologies, is one of the leading experts on HDLs. Mr. Zeidman has written two of the most popular textbooks on the subject, designed and sold tools that work with HDLs, designed many integrated circuits with HDLs, and given regular seminars on the subject. Had there been any mention of HDLs, it would have been apparent to him, but the topic description did not implicate HDLs in any way.

14. Zeidman Technologies filed an appeal with the Government Accounting Office. Zeidman Technologies was then contacted by Attorney Christopher Cole at the USAF Commercial Law & Litigation Directorate. Mr. Cole stated that he was looking into the issue.

15. On September 14, 2016, Mr. Cole sent Zeidman Technologies an email with a new and different explanation for the rejection. The new rejection states that Zeidman Technologies' proposal was not considered because "there is no mention of utilizing multiple processing cores efficiently, which is one of the main objectives of this solicitation." The rejection goes on to say that Zeidman Technologies did not address "likely future compute [*sic*] architectures (e.g., RADSPEED, MAESTRO, [Tilera], ARM, GPU, FPGA and X86 64)."

///

16. This new rejection was as baseless as the first one, because the evaluators had now changed the requirements from HDL, which was not mentioned anywhere in the RFP, to multicore processing, which is also not mentioned anywhere in the RFP. These are two completely different technologies that are unrelated.

17. Even so, Zeidman Technologies' proposal does specifically discuss multiprocessing as well as FPGAs, ARM processors. The proposal also discusses the fact that Zeidman Technologies' tool is "processor agnostic," allowing it to be used on any processor architecture in existence or in the future that has a C compiler. Zeidman Technologies actually did meet multicore processing criteria, even though the solicitation said nothing about multicore processing. It appears that Defendant did not even read Zeidman Technologies' proposal.

18. Mr. Cole informed Zeidman Technologies that Defendant granted two awards for proposals submitted for this topic, one to Colorado Engineering, Inc. (www.coloradoengineering.com) and another to EM Photonics, Inc. (www.emphotonics.com). Neither company displays any expertise in HDLs on its website, though Zeidman Technologies was originally rejected for not addressing HDLs, a subject where Zeidman Technologies has extensive expertise, and Robert Zeidman is considered a leading expert in the field. Colorado Engineering is a hardware design consulting firm that professes no particular software knowledge of any kind, let alone software for "utilizing multiple processing cores" for which the second rejection was given. EM Photonics does seem to develop software as well as hardware, but neither company discusses any expertise, as required by the proposal topic, in "accept[ing] C-language code for a suite of algorithms and output[ing] optimized code that can be compiled for a selected device," which is precisely what Zeidman Technologies' patented SynthOS tool does. Neither company expresses any knowledge of FPGAs, another supposed requirement of the proposal, where Zeidman Technologies has extensive knowledge and another area in which Robert Zeidman is considered a leading expert, having written a popular textbook and taught seminars on the subject.

///

///

19. After Zeidman Technologies pointed these things out to Mr. Cole, it received notice of Corrective Action from Defendant that states that the same people who evaluated Zeidman Technologies' proposal the first time would reevaluate its proposal and take "corrective action."

20. Defendant then issued entirely new reasons for the rejection. None of these stated reasons are valid, as Zeidman Technologies' proposal meets all of the stated criteria in the solicitation. Additionally, once again, the Defendant relied on unstated criteria in its evaluation, and ignored the evidence before them demonstrating Zeidman Technologies' obvious qualification for the solicitation.

BID PROTEST (28 U.S.C. § 1491)

21. Zeidman Technologies hereby incorporates by reference paragraphs 1 through 20, inclusive.

22. The Court of Federal Claims may grant the relief it considers proper, including declaratory and injunctive relief, where a protester succeeds in establishing prejudicial error.

23. Defendant has issued a procurement decision on the basis of unstated criteria, in violation of well-established law.

24. Defendant has issued a procurement decision without evaluating evidence presented in a proposal, in violation of well-established law.

25. The errors in the procurement process substantially prejudiced Zeidman Technologies' ability to obtain the contract at issue.

26. But for the errors in the procurement process, Zeidman Technologies would have had a substantial chance at obtaining the contract at issue.

27. Zeidman Technologies asks for declaratory judgment that the Defendant's procurement was based on unstated criteria, and failed to evaluate evidence presented in the proposal, therefore resulting in an arbitrary and capricious procurement decision, to Zeidman Technologies' detriment.

///

///

PRAYER FOR RELIEF

WHEREFORE, Zeidman Technologies prays for judgment against Defendant as set forth below:

1. For declaratory relief;
2. For injunctive relief;
3. For general damages in an amount according to proof;
4. For compensatory damages in an amount according to proof;
5. For attorneys' fees and costs to the fullest extent permissible by law; and
6. For any other and further relief that this Court may deem just and proper.

DATED: November 2, 2017

McMANIS FAULKNER

ELIZABETH PIPKIN

Attorney for Plaintiff,
Zeidman Technologies, Inc.

Exhibit A

Integrated Code Base and High Performance Embedded Computing Tool

Print

Agency:	Department of Defense
Branch:	Air Force
Program | Phase | Year:	SBIR | Phase I | 2016
Solicitation:	DoD 2016.1
Topic Number:	AF161-088

NOTE: The Solicitations and topics listed on this site are copies from the various SBIR agency solicitations and are not necessarily the latest and most up-to-date. For this reason, you should use the agency link listed below which will take you directly to the appropriate agency server where you can read the official version of this solicitation and download the appropriate forms and rules.

The official link for this solicitation is: http://www.acq.osd.mil/osbp/sbir/solicitations/sbir20161/index.shtml

Release Date:	Open Date:	Application Due Date:	Close Date:
December 10, 2015	January 11, 2016	February 17, 2016	February 17, 2016

Description:

TECHNOLOGY AREA(S): Electronics

The technology within this topic is restricted under the International Traffic in Arms Regulation (ITAR), 22 CFR Parts 120-130, which controls the export and import of defense-related material and services, including export of sensitive technical data, or the Export Administration Regulation (EAR), 15 CFR Parts 730-774, which controls dual use items. Offerors must disclose any proposed use of foreign nationals (FNs), their country(ies) of origin, the type of visa or work permit possessed, and the statement of work (SOW) tasks intended for accomplishment by the FN(s) in accordance with section 5.4.c.(8) of the solicitation and within the AF Component-specific instructions. Offerors are advised foreign nationals proposed to perform on this topic may be restricted due to the technical data under US Export Control Laws. Please direct questions to the AF SBIR/STTR Contracting Officer, Ms. Gail Nyikon, gail.nyikon@us.af.mil.

OBJECTIVE: Develop an integrated code base suite and a tool set that can generate high-performance, hardware platform-specific code.

DESCRIPTION: The Air Force and the space community need a modern space processor for missions in 2020-2030 that processes large volumes of data with sophisticated algorithms. Overhead Persistent Infrared (OPIR), radar, hyperspectral, and hyper-temporal concepts being studied today envision real-time computer systems for signal and image processing applications. They typically require low latency and high throughput of application processing, efficient utilization of system resources (compute, memory, bandwidth), low form factors (size, weight, and power demands), affordable software costs (code size, re-use, portability) and high autonomy. The Next-Generation Space Processor study funded by the Air Force Research Laboratory and NASA has evaluated architecture trades and the Space and Missile Systems Center is currently assessing a processor developed for the National Reconnaissance Office. Translating high level algorithm descriptions written in MATLAB to high performance and hardware efficient implementation remains challenging. High Performance Embedded Computing (HPEC) application processing almost always requires iterative rounds of software performance optimization to attain required application latency and throughput performance. Consequently, progress has been slow and comparison between architectures has been ambiguous, and this has precluded an informed decision to commit the considerable resources needed for design, qualification, and implementation of a particular architecture. This topic requests the development of a tool that accepts C-language code for a suite of algorithms and outputs optimized code that can be compiled for a selected device. In addition to making it easy and quick to port algorithms across different platforms for comparing the performance of different architectures, the tool should also allow software developers to concentrate on algorithmic advances rather than processor architecture peculiarities. Goals of 10X productivity improvement (e.g., through high-level abstraction). Performance speed-up based on platform tuning (e.g., cache re-sizing, core availability, internal bus performance, etc.) are also desired.

PHASE I: Identify/define hardware-aware optimization capes for likely future compute architectures (e.g., RADSPEED, MAESTRO [Tilera], ARM, GPU, FPGA and X86 64). Define exemplar data sets used to V&V implementation. Establish methodologies that support rapid platform-agnostic code generation capable of efficiently mapping algorithms to platform-specific features and exploiting available optimizations.

PHASE II: Create consolidated and integrated OPIR algorithm test suite from existing constituent algorithms; map high-level MATLAB algorithm specifications to related kernel and processing functional block implementations. Demonstrate ability to generate hardware platform-aware code using the consolidated and integrated OPIR algorithm test suite, execute the generated code on one or more of the likely future computer processor hardware platforms using the representative data sets.

PHASE III DUAL USE APPLICATIONS: Space qualify selected processor in 2020 timeframe.

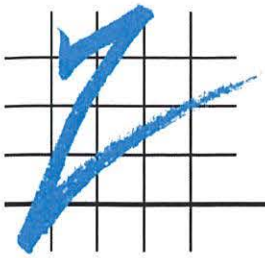
REFERENCES:

- John Keller, The future of high-performance embedded computing, August 11, 2014. Available online at http://www.militaryaerospace.com/articles/print/volume-25/issue-8/technology-focus/the-future-of-high-performance-embedded-computing.html.

- Commander's Handbook for Persistent Surveillance, Version 1.0. Available online at http://www.dtic.mil/doctrine/doctrine/jwfc/surveillance_hbk.pdf.
- DSP fact sheet. Online at http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=532.
- SBIRS fact sheet. Online at http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=22323

KEYWORDS: processor, efficient, processing, design, tool, HPEC, OPIR

Exhibit B



Zeidman Technologies

Department of Defense SBIR Proposal

Topic AF161-088

Proposal F161-088-1073

Topic: Integrated Code Base and High Performance Embedded Computing Tool
Technology Area: Electronics

Project Title: Synthesizing Application Specific Operating Systems (ASOS) for Present and Future Compute Architectures

Company:
Zeidman Technologies, Inc.
15565 Swiss Creek Lane
Cupertino, CA 95014

Principal Investigator: Jacob Harel

Table of Contents

Identification and Significance of the Problem or Opportunity, and Technical Approach

Embedded systems are found in everyday devices, from appliances to cars and from robots to aircrafts. Over the last few years a new category of embedded system has emerged, the Internet of Things (IoT). This new category describes objects and control systems that are connected together in an effort to provide a better level of connectivity and interaction. This IoT category includes wearable devices that will record the user during daily activities and home automation that will increase the integration of household appliances. All of these devices are connected to the cloud to enable remote control and to collect data that will optimize their performance.

SynthOS is a patented software tool that takes software tasks written by a developer and automatically generates a real-time embedded system using parameters set by the developer. SynthOS takes input code in the C programming language and produces output code in the C programming language. SynthOS can check for race conditions, deadlock situations, priority inversion, latency problems, and other hazards before the code is compiled. Options for developers currently include purchasing an off-the-shelf real-time operating system (RTOS) or writing a custom real-time scheduler from scratch. An off-the-shelf RTOS requires a large memory space, complex memory management hardware, and high-end microprocessors, which translates to higher system costs. Also, if existing edge devices do not already have the required hardware, they cannot be retrofitted with such an RTOS. Custom real-time schedulers are difficult to write and debug, and require developers to have significant expertise in real-time coding structures such as mutexes, semaphores, mailboxes, and message queues as well an understanding of hazards like deadlocks, race conditions, and priority inversion that can bring a system to a halt. SynthOS is a push-button solution that requires little real-time expertise and that creates solutions that are correct by design. SynthOS has been fully developed and tested, having successfully synthesized an application specific operating system (ASOS) for many different processors and platforms. It is also now available as a SaaS at www.SynthOSonline.com.

The aerospace industry can leverage tools for the IoT, and particularly SynthOS, to speed development of real-time computer systems for signal and image processing applications. Zeidman Technologies will use SynthOS, and its years of experience with real-time embedded systems, to apply this patented technology to support rapid platform-agnostic code generation capable of efficiently mapping algorithms to platform-specific features and exploiting available optimizations

Anticipated Public Benefits

The Internet of Things ("IoT") has already begun connecting everything from large, intelligent devices like supercomputers, personal computers, tablets, and smartphones to smaller devices like smart home applicances, home security systems, wearable devices, and smart light bulbs. This proliferation means that more software must be developed by more programmers. These embedded systems programmers require an RTOS to control the various tasks of the software. Roughly one quarter of all embedded software developers write their own operating systems[1]. Why is this so? Mainly because the off-the-shelf operating systems are overly complex for the majority of embedded systems being developed. These operating systems require a large memory space and complex processors, which translates to higher costs and higher power consumption. For high-end devices, this may not be a problem, but if we are to connect even the smallest devices, then every penny counts and longer operation time is an advantage. Off-the-shelf RTOSes also require a steep learning curve and specialized expertise, which is even more significant a requirement if a company develops its own RTOS in house.

Off-the-shelf RTOSes also have a problem with security. Because they are large and complex and designed to handle many different applications running simultaneously, they utilize task switching

[1] UBM – Embedded Market Study 4/21/2014, http://www.eetimes.com/document.asp?doc_id=1322014

mechanisms and message passing mechanisms that can be exploited by malware to corrupt the system. Any attacker can obtain a copy of the RTOS, reverse engineer it, test it, and find vulnerabilities such as buffer overflows, dangling pointers, race conditions, and priority inversions. The task of discovering and exploiting these vulnerabilities is made simpler by the fact that many of the RTOSes are open source and even most of the commercial RTOS vendors provide the source code.

There is a need for a tool that allows embedded programmers to generate operating systems automatically, but that still gives them full visibility to the code and full control over its development. This tool needs to address concerns of

- Time to Market
- Security
- Cost of goods sold
- Development costs
- Power consumption
- Portability
- Ease of maintenance

The pervasive IoT exacerbates this problem. IoT devices are often more resource limited than typical embedded systems. Also, they are usually designed by hardware engineers who have little experience writing an operating system. There needs to be a tool to automatically generate the RTOS according to the requirements of the tasks that the system needs to perform.

Such a tool would also allow the RTOS to be compiled as an application running on top of another operating system on a standard PC. If the RTOS application can access the hardware peripherals of the PC, the entire system could be debugged in software on an inexpensive platform before any hardware is even developed. This type of system emulation would substantially reduce system development time and significantly increase the reliability of the embedded system, two areas with which embedded system developers have struggled for many years. This simulation and testing capabilities will support modern development methodologies like Agile development, supporting quick and frequent development iterations.

With a great choice of platforms, processors, and interface devices, an embedded system designer would like to have a tool that optimizes the choice of platforms and hardware devices before committing to specific ones. A typical design process for an embedded system involves defining the hardware interfaces and writing drivers to support them. In many cases, even the specific processor is not important as long as the processor is available, meets the speed and cost constraints, and can be targeted by an available compiler. However, the majority of hardware interfaces are standard. In printed circuit board (PCB) level designs, the interfaces are often standard chips. In platform SOCs, the interfaces are usually IP cores. In either case, a library of drivers can be made available where each corresponds to different hardware interfaces. A tool is needed that could place a specific processor and the specific interfaces onto a board or piece of silicon without a great deal of hardware knowledge. Such a synthesis tool would mean that many more designers could design embedded systems and experiment with and optimize the hardware.

Description of the Technology

The patented technology from Zeidman Technologies that is used in SynthOS takes the concept of hardware synthesis and applies it to multitasking embedded software. SynthOS allows the software engineer to write code for each specific task in C. When one task needs to call another task, or wait for another task to complete, the software engineer inserts a special line of code that is recognized by SynthOS, called a "primitive." These primitives look just like simple function calls. The software

engineer also creates a simple project configuration file to specify the parameters of each task such as the task's priority, its relative frequency, the type of each task, and the interrupt mecahnism. SynthOS is then run on all of the task code, creating the appropriate semaphores and flags, and inserting appropriate task communication code at the appropriate points in the task. SynthOS also creates a real-time scheduler to manage the task. This process is illustrated in Figure 1.

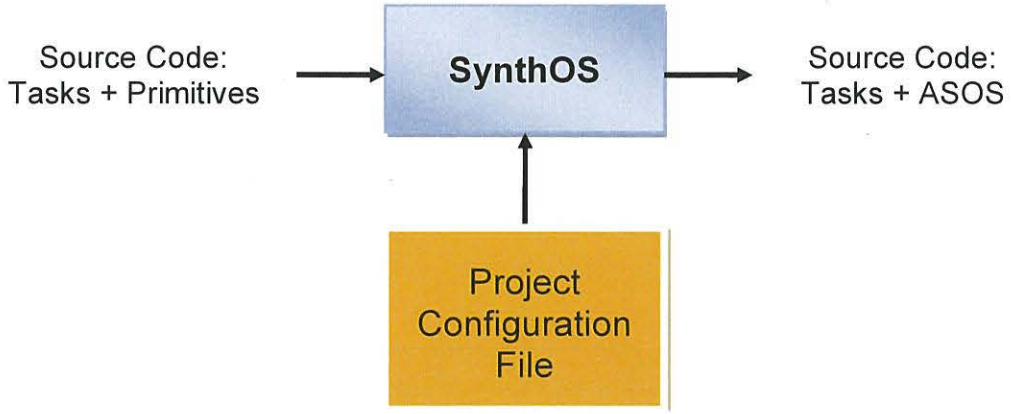


Figure 1. SynthOS code generation process

Examples of two resulting kinds of systems are shown in Figure 2. On the left side is a priority-based scheduler, while the right side shows a round robin scheduler. The specific scheduling algorithm is chosen by the developer in the project configuration file.

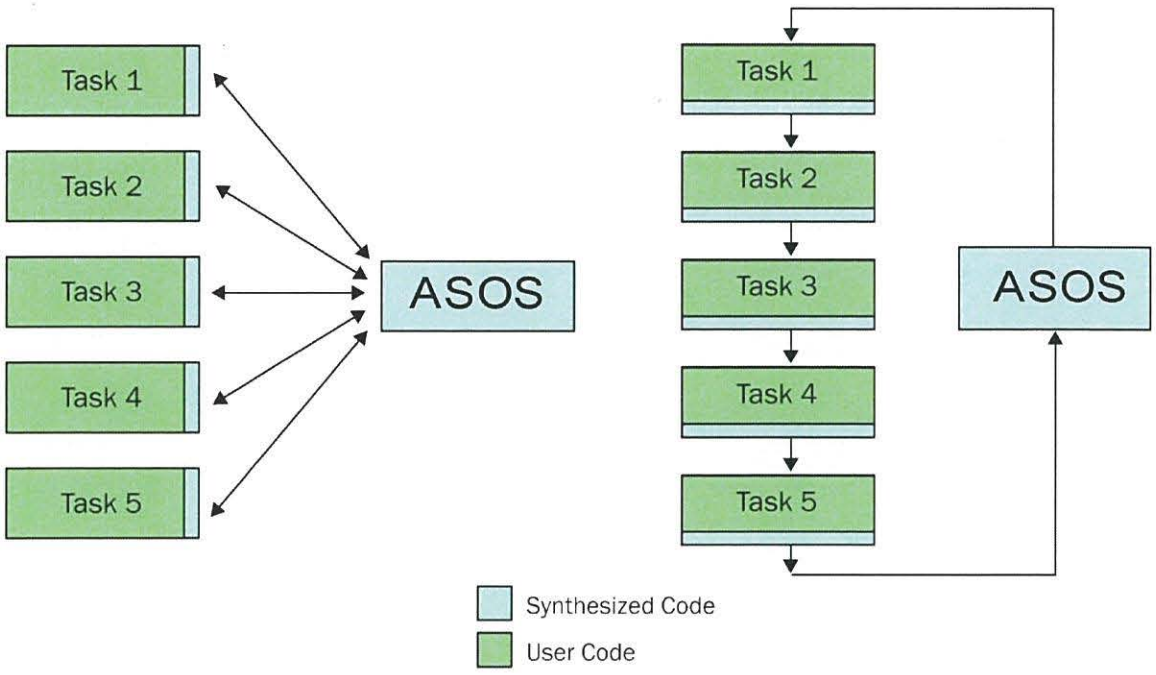


Figure 2. SynthOS code generation

Note that each task is mostly written by the user, but SynthOS inserts task control and communication code into each task. SynthOS also generates the ASOS that controls execution of each task. SynthOS can be tuned to different processors to allow it to optimize code for these processors. SynthOS automates the process of creating the operating system so that the engineer

can focus on writing the tasks that are specific to the devices. Because the output of SynthOS is in the same language as the input, the engineer has complete visibility to everything that is going on in the operating system. All of the tools that the engineer uses to compile and debug the tasks can also be used to compile and debug the ASOS. The SynthOS-generated code can also be easily modified by hand, if necessary, giving the engineer complete control over the code. A general diagram of the resulting system architecture is shown in Figure 3.

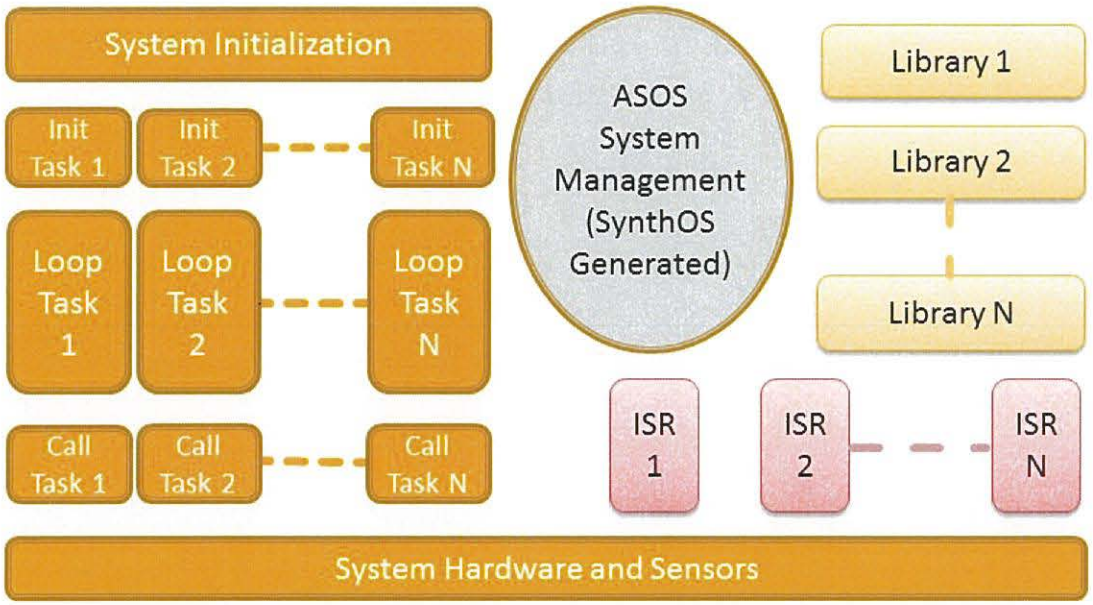


Figure 3. SynthOS-generated system architecture

Using SynthOS, there is no need to worry about race conditions, deadlocks, processor hogging, or unserviced tasks because the synthesized ASOS is correct-by-design. The resulting ASOS is portable to any processor that has a C compiler (virtually every processor) and is optimized for speed, size, latency times, and security. Furthermore, a system with a SynthOS-generated ASOS is more secure against malware than a system that uses an off-the-shelf RTOS for the following reasons:

- Malware cannot add security-breaching tasks to an existing system.
- Each ASOS is fully custom, so there is no way to examine an off-the-shelf system to discover its security flaws.

Zeidman Technologies, Inc. will develop a system for low latency and high throughput of application processing, efficient utilization of system resources (compute, memory, bandwidth), low form factors (size, weight, and power demands), affordable software costs (code size, re-use, portability) and high autonomy. The system will run on bare metal or on top of another OS, creating a virtual sandbox, isolating it from other applications on the system. We will reduce hardware costs and power consumption, compared to existing technologies, while increasing security and even allowing easy retrofitting of legacy systems.

Technical Objectives

This SBIR Phase I Project will identify and define hardware-aware optimization capes for likely future compute architectures (e.g., RADSPEED, MAESTRO [Tilera], ARM, GPU, FPGA and X86 64), define exemplar data sets used to verify and validate implementation, and establish

methodologies that support rapid platform-agnostic code generation capable of efficiently mapping algorithms to platform-specific features and exploiting available optimizations. Our objectives are to demonstrate a solution with the following characteristics:

1. **Low latency and high throughput of application processing**: SynthOS generates an ASOS that results in low latency and high throughput as demonstrated in other projects. **Technical question to be addressed**: Can a SynthOS-generated system produce low latency and high throughput for the specific applications and algorithms required for the Air Force and specifically lower latency and higher throughput than with off-the-shelf RTOSes.
2. **Efficient utilization of system resources**: SynthOS has been shown to generates an ASOS that utilizes system resources such as RAM and flash memory highly efficiently. In particular, SynthOS obtains more compute power and higher bandwidth from lower cost, processors while requiring less memory. **Technical question to be addressed**: Can a SynthOS-generated system efficiently utilize resources for the specific applications and algorithms required for the Air Force and specifically as compared with off-the-shelf RTOSes.
3. **Low form factors**: SynthOS has been shown to generates an ASOS that requires less hardware, smaller processors, and less memory, resulting in smaller size, weight, and power demands. **Technical question to be addressed**: Can a SynthOS-generated system result in lower form factor devices for the specific applications and algorithms required for the Air Force and specifically as compared with off-the-shelf RTOSes.
4. **Affordable software costs**: SynthOS is a low-cost solution compared to other tools or off-the-shelf RTOSes, and because the output code is in the C programming language, is it highly reusable and can even utilize compiled libraries written in other languages. Furthermore, because output is in C, all current compiler and debugging tools can be used without requiring the purchase of new development tools other than SynthOS. Many C tools are open source and thus have little or no cost.
5. **High autonomy**: Because SynthOS is a push-button solution, and required no expertise in RTOS issues, it gives the programmer a high degree of autonomy to focus on developing algorithms rather than being constrained by RTOS application specific interfaces (APIs).
6. **Platform independence**: SynthOS output code is in the C programming language, and thus a synthesized system is portable to any processor with a C compiler, which is almost any processor in existence. A further advantage is that the synthesized system is portable to any new processor as it comes on the market as long as it comes with a C compiler.
7. **Platform tuning**: Because SynthOS is portable to almost any processor, the resulting synthesized system can be run on different platforms to determine optimal performance. **Technical question to be addressed**: Which processor and which configuration produces the most efficient system for the specific applications and algorithms required for the Air Force.

Phase I Work Plan

The following plan describes the project development plan for demonstrating the capabilities of SynthOS for generating high-performance, hardware platform-specific code:

- **Task 1: Project specification**: With the help of Air Force engineers, define the specific set of signal and image processing applications to be implemented and synthesized into systems. Select a number of different hardware platforms for testing the system. Select a number of off-the-shelf RTOSes from comparison to the SynthOS-generated ASOS. Because not all RTOSes run on all processors, create a matrix of processors and RTOSes
- **Task 2: Test specification**: Define a suite of V & V tests to determine that the functionalities of these applications are correct. Define a suite of performance tests to determine the following characteristics:
    1. Latency

2. Throughput
3. Compute power
4. RAM memory usage
5. Flash memory usage
6. Power consumption
7. Bandwidth
8. Size
9. Weight

- **Task 3: Task code development**: Write code for each required application for the system, and combined applications, and completely test each task independently.
- **Task 4: ASOS generation and system implementation**: Use SynthOS to generate the application-specific operating system (ASOS) and any needed infrastructure for each application and combination of applications. Implement each system on each of the chosen hardware platforms specified in task 1. Run the V & V tests specified in task 2 on each platform, debug the software, and confirm that each system runs correctly on each platform.
- **Task 5: RTOS system implementation**: Use off-the-shelf RTOSes to create the system for each application and combination of applications. Implement each system on each of the chosen hardware platforms specified in task 1. Run the V & V tests specified in task 2 on each platform, debug the software, and confirm that each system runs correctly on each platform.
- **Task 6: System performance analysis and comparison**: Analyze the performance of each platform according to the characteristics specified in task 2 and compare each combination of processor, RTOS system, and ASOS system.

Phase I Performance Schedule

This is a preliminary schedule for the project. The overall project duration is 6 months.

Task #	Main People	Schedule (weeks)	Deliverables/Milestones
1	Bob Zeidman Jacob Harel Igor Serikov	2	Specification of tasks including applications, algorithms, software architecture, and flowchart.
2	Bob Zeidman Jacob Harel Igor Serikov	2	Specification of V & V tests.
3	Jacob Harel Igor Serikov	6	Tasks to implement exemplary algorithms and applications used by the Air Force.
4	Igor Serikov Jacob Harel	6	SynthOS configuration file and fully tested ASOS-based software system.
5	Igor Serikov Jacob Harel	6	Fully tested RTOS-based software systems.
6	Bob Zeidman Igor Serikov Jacob Harel	4	A document comparing system performance on various platforms with various operating systems.

Related Research or R&D

While Zeidman Technologies is the world leader in this new field of software synthesis for embedded systems, several similar projects are going on throughout the country. These programs are described below.

Ptolemy Project, University of California at Berkeley

The Ptolemy Project[2] at the University of California at Berkeley is a research project in the Center for Hybrid and Embedded Software Systems (CHESS) under the Department of Electrical Engineering and Computer Sciences. The project is directed by Professor Edward Lee, with whom we have been in contact though he has no official role in this proposal. The project is described on the Ptolemy website as follows:

> The Ptolemy project studies modeling, simulation, and design of concurrent, real-time, embedded systems. The focus is on assembly of concurrent components. The key underlying principle in the project is the use of well-defined models of computation that govern the interaction between components. A major problem area being addressed is the use of heterogeneous mixtures of models of computation.

The areas of the project that relate to our work in software synthesis include:

- **Cyber-physical systems**: Models of computation with time and concurrency, metaprogramming techniques, code generation and optimization, domain-specific languages, schedulability analysis, programming of sensor networks.
- **Understandable concurrency**: This effort focuses on models of concurrency in software that are more understandable and analyzable than the prevailing abstractions based on threads.
- **Semantics**: Domain polymorphism, behavioral type systems, meta-modeling of semantics, comparative models of computation.

Unikernels

Lately there has been an interest in unikernels, which are specialized, minimal operating systems that are customizable and run directly on a hypervisor or "bare metal" hardware. There have been papers written on the topic[3]. There is also an organization for unikernels, with a community site on the web at http://unikernel.org. Furthermore, the growing commercial interest in unikernels is demonstrated by the fact that the company Docker, Inc. recently purchased Unikernel Systems[4]. The ASOS that is produced by SynthOS is a unikernel.

Zeidman Technologies R&D

Zeidman Technologies has already successfully demonstrated the use of SynthOS to create custom systems on several projects, which are described below. We have also been issued seven patents that cover SynthOS technology as well as follow-on technologies, which are described below.

Previous Projects

SynthOS has been used to synthesize embedded software for a number of embedded systems including a multitasking webserver, a Lego Mindstorms robot, and a heterogeneous

---

[2] The Ptolemy Project, University of California at Berkeley, Center for Hybrid and Embedded Software Systems (CHESS) in the Department of Electrical Engineering and Computer Sciences, http://ptolemy.eecs.berkeley.edu

[3] Madhavapeddy, Anil and Scott, David J. *Unikernels: Rise of the Virtual Library Operating System*, ACM Distributed Computing magazine, Volume 11, Issue 11, January 12, 2014, http://queue.acm.org/detail.cfm?id=2566628

[4] Docker press release, January 21, 2016, https://www.docker.com/press-release-01212016docker-acquires-unikernel-systems-extend-breadth-docker-platfrom

multiprocessing robot arm. The webserver ran on an Altera Cyclone EP1C20 FPGA with an embedded NIOS 32-bit soft processor. The development for this very first application of SynthOS took several weeks, and the resulting ASOS was about 3K bytes. This same code was re-synthesized and compiled for an 8-bit Cypress PSoC resulting in an ASOS of about 1.2K bytes.

The Lego Mindstorms robot is shipped with 26K bytes of total memory. The operating system shipped with the robot uses about 22K bytes, leaving the user with only 4K bytes for applications. Many users substitute the widely available brickOS, which uses only about 11K bytes, leaving the user with 15K bytes for applications. SynthOS was used to synthesize a system for a Mindstorms robot with the resulting ASOS using only 2K bytes, leaving 24K bytes for user applications. SynthOS increased the user space by a factor of six and also supported more tasks and more global variables, allowing much more complexity for the applications.

A heterogeneous multiprocessing robot arm was controlled by a Xilinx Virtex II-Pro FPGA containing an embedded 32-bit PowerPC hard processor controlling a serial mouse and an embedded 32-bit MicroBlaze soft processor controlling a 5-joint robot arm, with each independent joint controlled by a separate task. The resulting ASOS for the PowerPC was 1.2K bytes while the ASOS for the MicroBlaze was less than 900 bytes. The robot arm is shown in Figure 4. The code for this robot demo can be downloaded from the Zeidman Technologies site at www.zeidman.biz/downloads.htm. A video of the robot can be viewed at https://youtu.be/F_juDOdzWRI.

Figure 4. Robot arm demo

We also created a multitasking robot based on an Arduino processor. The requirement was to build a robot that can move around an obstacle course while avoiding hitting walls and objects and not getting trapped in narrow sites. Furthermore, it was able to adjust the speed of its left and right tracks independently. One requirement was that if anything failed, such as one of the tracks getting stuck, it should give an indication (a beep) and shut down its power. We took an off-the-shelf robot kit, developed control algorithms, and used SynthOS to create an ASOS to schedule and coordinate the various robot tasks. The resulting robot is shown in Figure 5.

The use of the five simple SynthOS primitives enabled us to create a simple and straightforward architecture that included all the ASOS functionality to launch and synchronize the high-level tasks. The SynthOS tool also generated very efficient code, using only 1K byte of RAM and 13.5K bytes of flash memory including debug code. Taking out debug code reduced the RAM usage to 0.5K bytes and 8.6K bytes of flash memory. We ported the code to FreeRTOS, known as the most streamlined off-the-shelf RTOS available. The FreeRTOS solution required 1.25K bytes of RAM and 12.8K bytes of flash memory, significantly more than our SynthOS-based solution. This can be seen in Table 1. Less memory means not only lower costs, but also lower power consumption.

The code for the project can be downloaded at www.zeidman.biz/downloads.htm. A video of the robot can be seen at http://youtu.be/HzCGSk202gY.

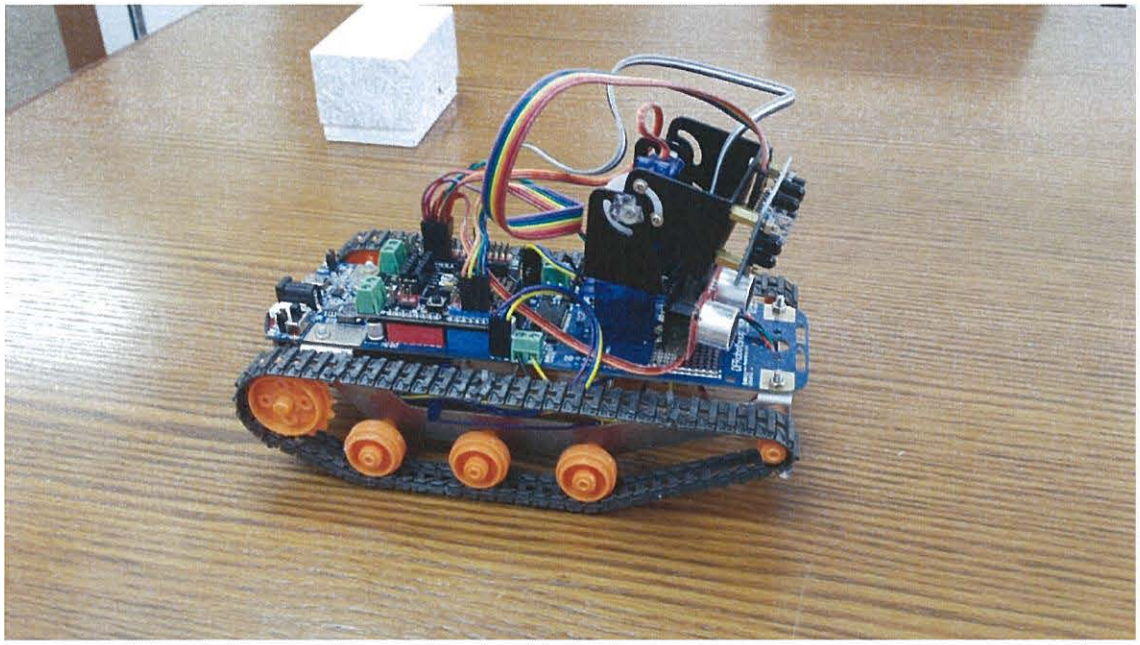


Figure 5. Autonomous robot demo

	RAM	Flash
FreeRTOS	1.25K	12.8K
SynthOS ASOS (with debug code)	1.00K	13.5K
SynthOS ASOS (without debug code)	0.5K	8.6K

Table 1. Comparison of FreeRTOS and SynthOS-generated ASOS

Patents

Zeidman Technologies has already filed and been granted seven patents, listed in Table 2, for the technologies of SynthOS and follow-on tools SynthEx, SynthEm, and SynthSys, described below.

SynthEx will take open-source drivers and other third-party code and automatically modify them for optimized use with SynthOS. SynthEx allows programmers to use the standard drivers and functions they need but use SynthOS to build an ASOS to run the routines, allowing them to leverage huge libraries of third party and open source code for use with a much smaller, faster, easier-to-build embedded system.

SynthEm allows an embedded system generated by SynthOS to be run as an application on an inexpensive, standard PC. SynthEm enables the system to use the PC hardware and peripherals directly. In this way, the PC is used to emulate the system, allowing testing, debugging, and functional optimization before the hardware is specified.

SynthSys takes the software description of the embedded system and synthesizes the hardware to implement the system by automatically selecting hard and soft intellectual property (IP) from an extensive library. Third-party tools then configure these hardware blocks, and connect them together to be synthesized, placed, and routed to create the platform on which to execute the software.

Patent
US 6,934,947: Visual Tool for Developing Real Time Task Management Code
US 7,647,583: Method And Apparatus For Emulating A Hardware/Software System Using A Computer
US 7,210,116: Method And Apparatus For Synthesizing A Hardware System From A Software Description
US 7,882,488: Software Tool For Synthesizing A Real-Time Operating System
US 7,900,187: Using Readily Available Driver And Application Source Code With A Synthesized Operating System
US 7,620,928: Method And Apparatus For Synthesizing A Hardware System From A Software Description
US 7,945,879: Method And Apparatus For Synthesizing A Hardware System From A Software Description

Table 2. Issued patents

Relationship with Future Research or Research and Development

If successful, this project will be able to demonstrate the advantages of embedded system synthesis and will provide a real-world application of this revolutionary new technology. It will show that intelligent software algorithms can create more efficient software designs while decreasing development costs, hardware costs, and power consumption. We intend to continue to optimize the synthesis capabilities and work on static analysis tools that can analyze software to find potential hazards and vulnerabilities before being deployed in the field or even loaded onto hardware. Our ultimate goal is to create a tool that, when given cost, power, size, and other constraints by the developer, can actually recommend the optimal hardware platform and then build that platform. We have already begun work on this strategy that will allow the algorithms to determine the firmware and hardware, essentially reversing the development process, letting the functionality determine the platform rather than the other way around. This project will be an excellent opportunity to start that effort that we can then expand in Phase II.

Commercialization Strategy

The Internet of Things ("IoT") has already begun connecting everything from large, intelligent devices like supercomputers, personal computers, tablets, and smartphones to smaller devices like smart toasters, home security systems, wearable devices, and smart light bulbs. This proliferation means that more software must be developed by more programmers. These embedded systems programmers require an RTOS to control the various tasks of the software. Roughly one quarter of all embedded software developers write their own operating systems because the off-the-shelf operating systems are overly complex for the majority of embedded systems being developed.

These operating systems require a large memory space and complex processors, which translates to higher costs and higher power consumption. For high-end devices, this may not be a problem, but if we are to connect even the smallest devices, then every penny counts and longer operation time is an advantage. Off-the-shelf RTOSes also require a steep learning curve and specialized expertise, which is even more significant a requirement if a company develops its own RTOS in house.

Off-the-shelf RTOSes also have a problem with security. Because they are large and complex and designed to handle many different applications running simultaneously, they utilize task switching mechanisms and message passing mechanisms that can be exploited by malware to corrupt the system. Any attacker can obtain a copy of the RTOS, reverse engineer it, test it, and find vulnerabilities such as buffer overflows, dangling pointers, race conditions, and priority inversions. The task of discovering and exploiting these vulnerabilities is made simpler by the fact that many of the RTOSes are open source and even most of the commercial RTOS vendors provide the source code.

There is a need for a tool that allows embedded programmers to generate operating systems automatically, but that still gives them full visibility to the code and full control over its development. This tool needs to address concerns of time to market, security, cost of goods sold, development costs, power consumption, and portability. SynthOS solves all of these problems and this project will give us credibility in the market.

Key Personnel

Principal Investigator

Jacob Harel, Vice President of Business Development/Product Management at Zeidman Technologies, is a high-tech entrepreneur and executive with over 25 years of industry experience in development and operation leadership with hands-on experience in software, hardware, and manufacturing. He was previously Senior VP of Operations & Engineering of Luidia and Engineering Manager at Electronics for Imaging (EFI). After completing his service in a technical unit of the Israeli Defense Forces, Jacob spent the early years of his career in hardware and software QA at Applied Materials and a variety of semiconductor fabrication equipment companies. He is the inventor on several patents. He holds a BSC in Computer Science and a BA in Economics from Tel Aviv University.

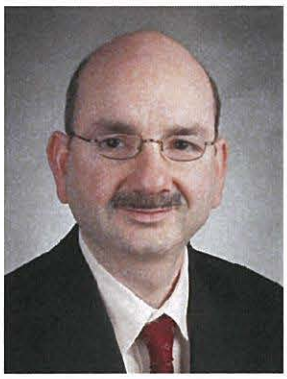
Investigator/Researcher/Programmer

Igor Serikov, Lead Developer at Zeidman Technologies, has devoted his career to the development of computer algorithms and software development tools. He has been a consultant to companies including Sygate, Symantec, NTT Docomo, Wyse Technology, and Ricoh Systems. Igor has a Bachelor's degree in computer science from National Taras Shevchenko University of Kyiv.

Investigator/Researcher

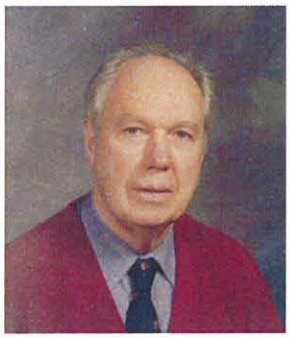
Bob Zeidman, President and CEO of Zeidman Technologies, is also the president and founder of Software Analysis and Forensic Engineering Corporation, the leading provider of software intellectual property analysis tools, having pioneered the field. He is also the president and founder of Zeidman Consulting, a premier contract research and development firm in Silicon Valley that provides engineering consulting to law firms about intellectual property disputes. Bob holds 22 patents, has written five textbooks, and was named the 2010 and 2015 Outstanding Engineer in a Specialized Field from the IEEE. He has a Master's degree in electrical engineering from Stanford University and two Bachelor's degrees, one in physics and one in electrical engineering, from Cornell University.

Technical Advisor



Kishore Manghnani has 25 years of management and marketing experience working at various systems and semiconductor companies in Silicon Valley. Kishore has built successful businesses from scratch in applications and technologies ranging from wireless micro-controllers for IoT, smart lighting, wireless home networking, broadband access, smart home, and other consumer electronics products. For the last 10 years, Kishore was Vice President & General Manager in the Communications Division of Marvell where he was responsible for the Smart Energy, IoT platform, and Wi-Fi business units. Prior to Marvell, Kishore had worked in various executive roles at Terayon, TeraLogic, LSI Logic, and Maxtor. He holds an MBA from Santa Clara University and an MSEE from the University of Hawaii.

Technical Advisor



Mike Flynn is Professor Emeritus of Electrical Engineering at Stanford University and Chairman of Maxeler Technologies. Mike has been teaching for over 40 years and is very well known and respected in the field of computer architecture. He has several textbook, over 250 papers, and is the recipient of many engineering awards including the ACM/IEEE Eckert-Mauchly Award for his technical contributions to computer and digital systems architecture. Mike has a Ph.D. in electrical engineering from Purdue University, a Master's degree in electrical engineering from Syracuse University, and a Bachelor's degree in electrical engineering from Manhattan College.

Facilities/Equipment

Zeidman Technologies leases an office in Cupertino, California where the majority of the research and development will take place. We already have software tools including compilers and debuggers and, of course, our SynthOS tool.

We will purchase hardware development boards with a variety of processors to test the software.

Consultants

No consultants are presently foreseen for the Phase I program. If a need should arise, we have connections with a number of consulting companies that specialize in hardware and software development.

Prior, Current or Pending Support

We have no prior or current support for a similar proposal. We have recently applied for an SBIR grant from the Department of Energy entitled "Convergent Smart Grid Communications and Application Architecture." The DOE project also involves synthesizing embedded systems using SynthOS, but the actual work is different from the work required for this project. We expect to hear whether we receive the grant in early May.

Cost Proposal

See attached.

References

1. Barr, M. *Programming Embedded Systems in C and C++*, Sebastopol, CA: O'Reilly and Associates, 1999.
2. Beal, J., Pianini, Danilo, and Viroli, Mirko. "Aggregate Programming for the Internet of Things," *Computer magazine*, Association for Computing Machinery, 9/13/2015, http://online.qmags.com/CMG0915/printpage.aspx?pg=24,25,26,27,28,29,30,31,32&pm=10
3. Berger, A., Hill, M., and Zeidman, B., "Software and RTOS synthesis: The next step in software development?" *Programmable Logic Design Line* (http://www.eetimes.com/design/programmable-logic/4015158/Software-and-RTOS-synthesis-The-next-step-in-software-development-), February 27, 2008.
4. Cataldo, A. "8-bit MCUs chase high-end features," *Embedded.com*, http://www.eetimes.com/document.asp?doc_id=1147994, Dec 2 2003.
5. *Docker press release*, January 21, 2016, https://www.docker.com/press-release-01212016docker-acquires-unikernel-systems-extend-breadth-docker-platfrom
6. Harel, J. and Serikov, I., An Application Specific Operating System for the Kinoma Create, Embedded Computing Design, 12/29/2015, http://embedded-computing.com/articles/an-system-the-kinoma-create.
7. Kang, D.I., Gerber, R., Golubchik, L., Hollingsworth, J.K., and Saksena, M. "A Software Synthesis Tool for Distributed Embedded System Design," Association for Computing Machinery, 1999, http://www.cs.umd.edu/~hollings/papers/lctes99.pdf.
8. Kantee, A. "The Rise and Fall of the Operating System," *;login:*, October 2015 Vol. 40, No. 5, www.usenix.org.
9. Lee, E. A. "Embedded Software – An Agenda for Research," University of California at Berkeley, UCB ERL Memorandum M99/63, 12/15/1999.

10. Leung, I. "MCUs face price pressures, migration to 32-bit and ARM," *Electronics News*, April 26, 2013, http://www.electronicsnews.com.au/news/mcus-face-price-pressures-migration-to-32-bit-and.
11. Nacul, A. C. and Givargis, T. "Lightweight Multitasking Support for Embedded Systems using the Phantom Serializing Compiler," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*.
12. Madhavapeddy, A. and Scott, D. J. *Unikernels: Rise of the Virtual Library Operating System*, ACM Distributed Computing magazine, Volume 11, Issue 11, January 12, 2014, http://queue.acm.org/detail.cfm?id=2566628
13. Proffitt, B. *How Big The Internet Of Things Could Become*, http://readwrite.com/2013/09/30/how-big-the-internet-of-things-could-become#feed=/author/brian-proffitt&awesm=~oFQQrdzaXPcdJc, September 30, 2013.
14. Proffitt, B. *The Internet Of Things In 2014 - Steady As It Goes*, http://readwrite.com/2013/12/27/2014-will-see-small-moves-towards-internet-of-things#feed=/author/brian-proffitt&awesm=~oFQQEFUAPYhg91, December 27, 2013.
15. Sardana, S. "Is There An App For That? How The 'Internet Of Things' Is Changing The Consumer Device Landscape," *Forbes*, May 29, 2014, http://www.forbes.com/sites/sanjeevsardana/2014/05/29/is-there-an-app-for-that-how-the-internet-of-things-is-changing-the-consumer-device-landscape.
16. *SynthOS Users Guide*, Cupertino, CA: Zeidman Technologies, Inc., 2014, http://zeidman.biz/documents/SynthOS_Users_Guide.pdf.
17. *The Ptolemy Project*, University of California at Berkeley, Center for Hybrid and Embedded Software Systems (CHESS) in the Department of Electrical Engineering and Computer Sciences, http://ptolemy.eecs.berkeley.edu
18. *UBM – Embedded Market Study,* 4/21/2014, http://www.eetimes.com/document.asp?doc_id=1322014.
19. Wong, Bill. "Development Tools Move To The Cloud," *Electronic Design*, http://electronicdesign.com/print/dev-tools/development-tools-move-cloud, January 24, 2014.
20. Zeidman, Bob, "ASOS: A new software development paradigm for the Internet of Things" Embedded.com, http://www.embedded.com/design/operating-systems/4431826/NEW--ASOS--A-new-software-development-paradigm-for-the-Internet-of-Things---Part-1--Basic-building-blocks and http://www.embedded.com/design/operating-systems/4431775/2/ASOS--A-new-RTOS-paradigm-for-the-Internet-of-Things---Part-2--Building-a-project-file
21. Zeidman, B., "How to choose an RTOS for your FPGA and ASIC designs," *Programmable Logic Design Line* (http://www.eetimes.com/design/programmable-logic/4015109/How-to-choose-an-RTOS-for-your-FPGA-and-ASIC-designs), May 10, 2007.
22. Zeidman, B., "Software Synthesis for Embedded Systems," Silicon Valley Code Camp, October 8, 2006.
23. Zeidman, B., "Using software synthesis for multiprocessor OS and software development," *Embedded.com* (http://www.embedded.com/design/prototyping-and-development/4006506/Using-software-synthesis-for-multiprocessor-OS-and-software-development.
24. Zeidman, B., "Software Synthesis for OS-Independent Coding," *Dr. Dobb's Journal* (http://www.drdobbs.com/architecture-and-design/software-synthesis-for-os-independent-co/184406036.
25. Zeidman, B., "RTOS Synthesis for Embedded Systems," Server Blade Summit, March 24, 2005.

26. Zeidman, B., "RTOS Synthesis to Reduce Power Consumption," DesignCon 2005, February 1, 2005.
27. Zeidman, B., "Software synthesis for embedded systems," *Embedded Systems Programming* (http://www.embedded.com/design/prototyping-and-development/4006439/Software-synthesis-for-embedded-systems) February 2005, pp 36-43.
28. Zeidman, B., "Roll Your Own Real-Time OS," *BladeLetter*, Q3 2004, p. 8.
29. Zeidman, B., "Software synthesis is productive for system design," *EE Times* (http://www.eetimes.com/document.asp?doc_id=1148219), January 12, 2004.
30. Zimmer, M., Broman, D., Shaver, C., and Lee, E. A. "FlexPRET: A Processor Platform for Mixed-Criticality Systems." *Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS)*, Berlin, Germany, April 15-17, 2014.