Practical Evaluation of Synthesis Performance on the BeagleBone Black

Ivan Franco
Input Devices and Music Interaction Lab
CIRMMT, McGill University
Montreal, QC, Canada
ivan.franco@mail.mcgill.ca

Marcelo M. Wanderley
Input Devices and Music Interaction Lab
CIRMMT, McGill University
Montreal, QC, Canada
marcelo.wanderley@mcgill.ca

ABSTRACT

The proliferation and easy access to a new breed of ARM-based single-board computers has promoted an increased usage of these platforms in the creation of self-contained Digital Music Instruments. These directly incorporate all of the necessary processing power for tasks such as sensor signal acquisition, control data processing and audio synthesis. They can also run full Linux operating systems, through which domain-specific languages for audio computing facilitate a low entry barrier for the community.

In computer music the adoption of these computing platforms will naturally depend on their ability to withstand the demanding computing tasks associated to high-quality audio synthesis. In the context of computer music practice there are few reports about this quantification for practical purposes. This paper aims at presenting the results of performance tests of SuperCollider running on the BeagleBone Black, a popular mid-tier single-board computer, while performing commonly used audio synthesis techniques.

Author Keywords

Embedded systems, Sound Synthesis, BeagleBone Black, SuperCollider

ACM Classification

C.5.3 [Computer System Implementation] Microcomputers — Microprocessors, D.2.8 [Software Engineering] Metrics—Performance Measures, J.5 [Computer Applications] Arts and Humanities

1. INTRODUCTION

Many Digital Music Instruments (DMI) rely on system architectures composed by two main elements: the controller and the computer. The controller is responsible for capturing information about the instrumental gesture of the performer and sending control data to the computer, which in turn executes the required sound synthesis and mapping processes [10].

But as DMI developers further advance the knowledge of underlying subjects like sound synthesis, computer-aided composition or human-computer interaction models, a fundamental goal remains: how can these new instruments provide the same gratifying immediacy and musical expressiveness as their acoustic counterparts?

A new breed of miniaturized embeddable systems grants the ability to rethink DMI architectures and consider the advantages of integrating both elements, by incorporating all the necessary components for the control and production of sound into body of the instrument.

Some possible advantages of integrated designs have already been a topic of discussion among DMI developers. Longevity is one of the strongest arguments, since the ability to have dedicated processors, that would not need to be stressed by having to perform day-to-day computing tasks, could help prevent performance degradation or software obsolescence [3]. Another relevant feature of self-contained instruments is that they do not require a previous setup between controller and computer, which often results in a need for continuous checking of connections (physical and virtual) and more importantly mapping algorithms. This often tedious and laborious task, without which the instrument cannot be played, might detract from the immediacy found in other types of musical instruments. Self-contained instruments also have the potential to offer more focused and contextualized functionality, which could significantly boost ease-of-use and still provide less-tech savvy users with the access to idiosyncratic aspects of computer music. As noted by Berdahl and Chafe, self-contained instruments are "complete and independent of external computer systems, so they can be operable for a longer period of time and can be demonstrated at a moment's notice" [2]. But the most obvious drawback in the use of embedded computing in DMIs is that these small computers will naturally have less processing power than regular computers. Thus it would be relevant to try to quantify their performance in order to better understand if they are appropriate for the desired goals of sound generation and processing.

2. EMBEDDED COMPUTING

In the last twenty years an appreciable number of new computing platforms have emerged with the goal of supporting innovation in the hobbyist and do-it-yourself communities, granting NIME developers and researchers access to prototyping technologies that otherwise would be confined to industrial production. The first wave of these products was associated to the democratization of micro-controllers, starting with Parallax's BASIC Stamp and later the Arduino, which has now been in the market for about ten years. The overwhelming success of these technologies might be mainly due to their ease-of-use and low-entry barriers. Non-professional developers could now write code in simplified high-level languages, like Basic or Wiring, and program micro-controllers by simply connecting them directly to personal computers, without the need for complex hardware programmers or compiling toolchains.

We are now observing a rapid emergence of a second wave of embedded computing platforms, related to the proliferation of systems-on-chip (SoC) and their application in single-board computers (SBC). The recent boost of performance of ARM-based SoCs is derived from the growth of the mobile computing industry and the need to reach relatively high processing power in reduced footprints, while simultaneously targeting extremely low power consumptions. We believe these mobile-oriented processors will eventually substitute the ones found in today's desktop and laptop computers targeted at the more casual user. In broad terms we can say that the same computing power of the desktop computer of 10 years ago now fits any pocket.

3. ARM SINGLE-BOARD COMPUTERS

Although most new mobile devices offer novel presentation layers and interaction paradigms, through their touch-focused user interfaces and context-aware applications, behind their external appearance most of them rely on traditional operating systems, generally some variant of Unix. Many single-board computers based on these newer SoCs are capable of performing all of the functionalities expected from the traditional computer, providing connectivity to additional peripherals like controllers, screens or mass storage, often through standard protocols and connectors.

Although ARM processors have considerably different architectures from X86 processors, current compilers are well adapted and mature enough so that using computer code that had originally been written for Linux and X86 processors becomes quite straightforward. The ability for these devices to run well established domain-specific software tools and libraries, which have been in development for a long time, maturing to become *de-facto* standards in determined communities, is a big advantage over past embedded computing architectures that normally demanded dedicated implementations, many using much lower-level code.

Many of these SBCs also provide direct access to the microprocessor chip, mapped to a General-Purpose Input / Output (GPIO) socket, through which many of the pins can be exposed for additional tasks, like digital communication to other integrated circuits or analog signal acquisition. Essentially this type of morphology grants these computers with the same type of expansion found in micro-controller development boards like the Arduino, in which integration with more more elaborate electronics setups or special purpose daughter boards becomes relatively easy to accomplish.

This easy portability of well-established software, coupled with general-purpose digital and analog I / O, a small footprint and low power consumption grants these ARM-based SBCs with most of the necessary features to suggest designs where they could become the main and only processor for a Digital Music Instrument. Indeed these new pocket-sized computers have already fueled interest from the NIME community, since these characteristics pave the way for the creation of new breeds of self-contained digital music instruments.

4. PREVIOUS WORK

The SoundLab [6] is one of the first attempts at developing a fully portable and auto-sufficient DSP unit. The system was created by Steve Curtin in 1994, during a residence at STEIM. SoundLab represented a natural evolution from SensorLab, a previously existing portable signal acquisition system, that could be clipped to a belt, thereby providing more freedom of movement on stage.

The Gluiph [8], developed in 2003 by Kartadinata, was one of the earliest implementations of a base system for integrated DMIs using single-board computers. Kartadinata mentions that his motivation for creating a self-contained instrument was to foment "mature instruments with a stronger identity," surpassing the fragmented nature of most live electronic music setups. The Gluiph system was based on a custom board developed by Mtronix, powered by a Philips TriMedia CPU. Instead of using custom software, and in order to achieve maximum flexibility, the instrument was programmed using a port of Pure Data.

In Fungible Interfaces[7], Hollinger, Thibodeau and Wanderley describe the implementation of an embedded hardware platform for standalone instruments, emphasizing the advantages of easy management, portability and resilience of instruments that are untethered from general-purpose computers. Their system was based on a Programmable System-on-Chip (PSoC) developed by Cypress, featuring an additional ARM7 processor (LPC2468) for control.

More recently there has been a growing number of examples of self-contained instruments that use off-the-shelf and easily accessible ARM-based SBCs. Such instruments include the creations by CCRMA students using Satellite CCRMA [3], a ready-to-use Linux distribution that can be easily installed in SBCs like the Raspberry Pi and the Beagleboard XM. Two instruments that are based on this system are the Sound Flinger [4], an haptic spatializer in the form of a box with four motorized sliders positioned at its edges and the Deckle [5], an electroacoustic drawing board with embedded piezo microphones, using the sound of the scribbling on the board's surface for cross-synthesis. Satellite CCRMA has also been extensively used in the creation of small dedicated effects stomp boxes, including granulators, pitch-driven synths or more traditional guitar effects like wah-wahs, fuzzes or octave dividers. CCRMA has promoted summer workshops specially aimed at building these customized stomp boxes.

Other examples include the El-Lamellophone [12], another hyperinstrument that uses piezo-electric pickups mounted on the underside of the instrument's body for direct audio acquisition and processing and Range [9], an autonomous processing unit for guitar effects, with pitch tracking and several potentiometers for parameter control. Both these implementations are based on the BeagleBone Black, another popular ARM-based SBC. The authors of Range refer the high level of autonomy of their system and the versatility that domain-specific languages like Pure Data provide, with a "wide range of possibilities and for both digital audio effects and their control." [9]

5. SYNTHESIS PERFORMANCE EVALUATION

Although the previous examples prove that it is somewhat possible to use these types of computers in the construction of DMIs, there is generally little information about their performance.

Previous work on the evaluation of the BeagleBone Black as a viable computing platform for digital music instruments centered mainly on the topic of latency, with reports by both MacConnell [9] and Topliss [11] proving the ability for this SBC to deliver an acceptable range of audio latencies between 2.3 and 12 ms.

But an important aspect that remains to be studied is the capability for these single-board computers to support the considerable amount of processing power that elaborate sound synthesis might comprise. Typically the only consistent benchmark used to categorize audio synthesis capabilities is the shear number of oscillators that a determined system can play simultaneously. Although this technique might offer one possible method of comparing system performance, it is not very informative for the user, since it does not help in the comprehension of how that measure might translate into other modern synthesis techniques (apart from additive synthesis). Benchmarking could possibly be more informative by similarly counting the number of voices that can be reproduced simultaneously but including other synthesis techniques, like granular synthesis or the phase vocoder. Thus we set out to perform a series of tests that provide a better indication of the expected performance from a SBC like the Beaglebone Black.

5.1 Test Conditions

Performance tests of this nature are greatly affected by the choices of operating system, synthesis engine or the resolution of the sound processing calculation. In the next section we describe the conditions used in this performance test.

5.1.1 The BeagleBone Black

We chose the BeagleBone Black (BBB), due to its popularity and wide availability. The BBB is a 3.4" by 2.1" ARM SBC with the following specifications:

- 1 GHz ARM® Cortex-A8 processor
- 512 MB DDR3 RAM
- 4GB eMMC on-board flash storage
- USB host
- Ethernet
- HDMI
- GPIO on 2x 46 pin headers

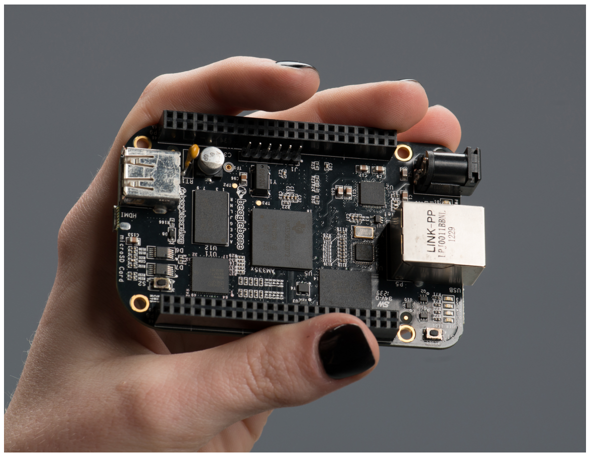

Figure 1: The BeagleBone Black (photo by Adafruit Industries / CC BY)

5.1.2 Software and Audio Input / Output

Initially the BBB was shipped with a custom Linux distribution named Angstrom, recently abandoned mainly due to the faster progress of other alternative distributions. One of these is Debian, that offers an ARM version with full support for ARM's hard floating point engine, which lead it to eventually becoming the officially supported OS for the BBB.

After the OS installation to the on-board EMMC we compiled SuperCollider 3.7, using gcc with instructions to use hard floating point and NEON, ARM's Single Data Multiple Instruction (SIMD) engine. The corresponding gcc flags would be:

```
  -march=armv7-a -mtune=cortex-a8 -mfloat-abi=hard -
mfpu=neon
```

We opted for SuperCollider for the synthesis engine since it is among the most popular languages for computer music and is entirely text-based, so that it could be used with a headless version of Debian.

Audio I / O was done by connecting an external USB soundcard, with 16 bit depth and a sampling rate of 44.1 kHz. SuperCollider's audio was channeled through JACK [1], a Linux API for inter-application audio routing. The vector size used was of 512 samples.

5.1.3 Sound Synthesis

Next we chose the following set of synthesis techniques, to be tested under the previously described conditions:

- Wavetable Synthesis, with an arbitrary table of 512 samples, which can can be dynamically modified at will.
- Granular Synthesis, with each grain duration of 10 ms and triggered periodically at a rate of 100 Hz.
- Cross-synthesis 1, consisting on the convolution between a sawtooth oscillator and a *fixed kernel* with a FFT frame size of 512 samples.
- Cross-synthesis 2, consisting on the convolution between a sawtooth oscillator and a *dynamic kernel* with a FFT frame size of 512 samples.
- Phase Vocoder Pitch Shifter, through FFT bin shifting and a frame size of 512 samples.

All the programmed synthesizers had a similar structure, with an envelope per voice for amplitude control and stereo channel expansion by duplicating the signal at the output. In practice each voice corresponded to the instantiation of a synth. A simplistic code example of a synth definition that would follow the described structure would be equivalent to:

```
SynthDef(\wavetable, {| outbus = 0, freq = 440, gate
= 1, bufnum = 1 |
  var source, env, out;
  source = Osc.ar(bufnum,freq);
  env = EnvGen.kr(Env.adsr(0.01, 0.5, 0.5, 0.1), gate,
doneAction:  2);
  out = source * env;
  Out.ar(outbus, [out, out]);
})
```

These definitions can then be used to spawn synths by sending OSC commands to the SuperCollider server, which in turn will instantiate and output an independent voice.

An increasing number of sustained voices were played, until there was any noticeable CPU overload, resulting in audible interruptions or artifacts. Additionally, we also monitored the overload point by observing any error logs from the JACK server. Another shell session was used to measure the percentage of CPU load through the Linux native processing monitor *top*.

Table 1: Sound Synthesis Test Results

Synthesis method	Number of voices
Wavetable	184
Granular	26
Cross-synthesis 1	20
Cross-synthesis 2	12
Phase Vocoder (pitch shifter)	16

5.2 Test Results

Table 1 shows the results from the performance tests of the various synthesis methods, with the corresponding number of possible simultaneous voices. In all of the cases the corresponding CPU maximum utilization by the SuperCollider server that resulted in audible artifacts was registered at an average of 84%. As expected the total number of voices diminishes with the complexity of the chosen synthesis. Wavetable synthesis can be considered extremely cheap, with about 10 times more possible voices when compared to the remaining methods. All others can still accomplish more than 10 voices, achieving a polyphony equivalent to harmonic articulation possible in instruments like the keyboard. Since the voices (in this case individual synths) can be freed after a note release, as long as the corresponding release times are not excessive, it should be possible to avoid processing overload. In any case, strategies for voice stealing can always be implemented by releasing older notes when a certain established threshold in number of active voices is surpassed.

6. CONCLUSIONS

This paper presented an informal strategy for estimating the processing power of the BeagleBone Black when performing several different synthesis methods, which should help in a practical evaluation of its application in the construction of self-contained Digital Music Instruments.

The choices of synthesis methods and audio engine were based on the assumption that they represent commonly adopted practices, thus providing better terms of comparison to other computing systems. In the case of SuperCollider, it is important to refer this software trades efficiency for the convenience of an easily-programmable virtual machine. Better results could be expected if the implementation was done using a lower level programming language like C, which could then be compiled using ARM-optimized toolchains. Still we believe these tests to be more useful in the context of a practical use by the computer music community, which tends to favor the ease-of-use and high-level functionality of domain-specific languages that provide a consistent and mature code base. Another advantage of using interpreted languages is that the effort of porting a determined implementation to different hardware systems is only conditioned by the ability to recompile the synthesis engine.

The registered results demonstrate that the BeagleBone Black is a single-board computer capable of supporting sound synthesis. Furthermore the BBB has a market cost of approximately $US 55, a viable choice for a system that could be fully dedicated to a single DMI. It is expected that this performance / price ratio will increase in the next years, with newer ARM-based boards already announced, like the next generation Beagleboard-X15 that will have a dual Core A15 processor running at 1.5 GHz and 2GB of DDR3L memory. Other alternatives in the market include the Odroid-U3, with a powerful quad-core ARM A9 processor, while maintaining an affordable price of $US 69. Some of these boards also include co-processors, like the Programmable Realtime Unit SubSystem (PRUSS) in the case of the BeagleBone Black or the C66x DSPs on the upcoming Beagleboard-X15. These co-processors could be used to increase performance in input / output of audio and control signals or serve as processing boosters in DSP computing.

7. ACKNOWLEDGMENTS

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for an individual Discovery Grant.

8. REFERENCES

[1] JACK Audio Connection Kit.

[2] E. Berdahl and C. Chafe. Autonomous New Media Artefacts (AutoNMA). In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 322–323, Oslo, Norway, 2011.

[3] E. Berdahl and W. Ju. Satellite CCRMA: A musical interaction and sound synthesis platform. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 173–178, Oslo, Norway, 2011.

[4] C. Carlson, E. Marschner, and H. McCurry. The sound flinger: A haptic spatializer. In *Proceedings of International Conference on New Interfaces for Musical Expression*, pages 138–139, 2011.

[5] H. Choi, J. Granzow, and J. Sadler. The Deckle Project : A Sketch of Three Sensors. In *Proceedings of International Conference on New Interfaces for Musical Expression*, pages 512–515, Michigan, United States, 2012.

[6] S. Curtin. The SoundLab: a wearable computer music instrument. In *Proceedings of the International Computer Music Conference*, Aarhus, Denmark, 1994.

[7] A. Hollinger, J. Thibodeau, and M. M. Wanderley. An embedded hardware platform for Fungible Interfaces. In *Proceedings of the International Computer Music Conference*, pages 26–29, 2010.

[8] S. Kartadinata. The Gluiph: A Nucleus for Integrated Instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 180–183, Montreal, Canada, 2003.

[9] D. MacConnell, S. Trail, G. Tzanetakis, P. Driessen, and W. Page. Reconfigurable autonomous novel guitar effects (range). In *Proceedings of the International Conference on Sound and Music Computing*, Stockholm, Sweden, 2013.

[10] E. R. Miranda and M. M. Wanderley. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. A-R Editions, Inc., 2006.

[11] J. W. Topliss, V. Zappi, and A. McPherson. Latency Performance for Real-Time Audio on BeagleBone Black. In *International Linux Audio Conference*, Karlsruhe, Germany, 2014.

[12] S. Trail, D. MacConnell, L. Jenkins, J. Snyder, G. Tzanetakis, and P. Driessen. El-Lamellophone - A Low-cost, DIY, Open Framework for Acoustic Lemellophone Based Hyperinstruments. In *Proceedings of International Conference on New Interfaces for Musical Expression*, pages 537–540, London, United Kingdom, 2014.