Gem5 Compatibility with Buildroot

Department of Computer Science Charleston Southern University
CSCI 540 Advance Computer Architecture
Jordan Kendale Manier
Final Research Paper

Abstract

Buildroot is highly configurable method used of constructing operating systems. Buildroots compatibility with Gem5 is what we explored in the research conducted. Buildroot's automated system cuts the time to create a bootable image considerably. In our research we configured and compiled two separate processor architectures, ARM & AARCH64. We had a failed attempt to compile x86_64 processor. This paper outlines the work conducted to test the compatibility between buildroot and Gem5.

1 Introduction

Innovation of multicore system and expanding caches levels are causing architecture for systems to be more complex. These architectures are highly configurable, the combination of configurations is near endless. When testing and researching different architectures, simulations if these configurations would aid in the ability to determine which is optimal. Gem5 offers the ability to assist researchers with simulating the different, complex architectures. Like the expanding architecture variants. Gem5 [1] offers an extremely moldable, modular simulation system that has the ability of evaluating a wide range of systems. Gem5 provides the ability to be flexible with its configuration by offering a variety of CPU, memory and system execution models.

Simulation Is defined as the art of imitation. Simulations could possibly be used in every field of research. The ability to simulate a process, event, or system proves to be very useful. The ability to simulate offers the means to save funds, time and resources. Simulations are merely an abstraction of reality. This abstraction is conducted in extreme detail. Being as accurate as possible will allow for a more accurate simulation. In other cases, simulations can be used to focus on one area of a system. There are three different of simulations: Live simulations, Virtual simulations, and Constructive simulations. These three different types of simulation can be combined together in one case. Simulations take a broad field in. Simulation can be used in behavioral experiments, human interactions, animal interactions, technology reason and even environmental research. Live simulations often involve human and their interaction in a live setting or event. A simulation of a football game scenario would be included as a live simulation. This simulation could imitate a game scenario with players practicing for a game.

Virtual simulations also involve humans but in a virtual setting. This includes a simulation of a videogame with a virtual computer-controlled equipment. The computer controls the simulation for the human. Constructive simulations in most cases don't involve humans. Constructive simulation is time driven, and the sequencing of events. This simulation could be used to determine the weather pattern for a week. In the case of this research simulation is used in a virtual setting. Simulating different components of a computer architecture.

Buildroot [2] was designed to use cross-compilation in order to automate and simplify the process of building a complete Linux system for an embedded system.

Buidlroot does this by generating a cross-compilation toolchain, Linux Kernel image a bootloader for your target and a root filesystem. Buildroot was designed to be highly configurable also. This is because of the variety of embedded linux system. This system solves the issue of creating a bootable image for architectures. Buildroot's menu driven configuration allows the ability for easy selection of chosen configurations. The only issue is determining the choice of configuration.

2 Architectures

In the research we conducted, we attempted to construct several different operating system (OS) images. A few were successful, and others weren't. The successful images include: ARM [3] and AARCH64 [4] processors. These two images were bootable with Gem5 in full simulation mode. Figure 1 contains a screenshot of the ARM image finish statistics. 128 minutes to be configured and build in gem5.

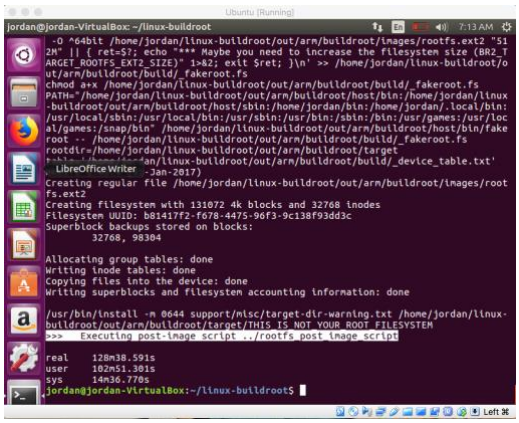

Figure 1, ARM final build with Gem5

The second image, AARCH64 was also bootable in Gem5 see Figure 2. Both images contained PARSEC [5] and Splash2x [6] benchmarks. In the Attempt to compile x86_64 [7] several errors occurred, and we were ultimately unable to successfully boot this image in gem5.

2.1 ARM/AARCH64

ARM processor is a CPUs based on the RISC (reduced instruction set computer) architecture developed by Advanced RISC Machines (ARM). ARM makes two RISC [8] multi-core processors, 32bit and 64bit. The RISC CPU's were created to perform a smaller number of the different types of computer instructions. This was done to allow RISC processors to operate at a higher speed, thus performing more million if instructions per second. RISC CPU's outperform by stripping away unnecessary instructions and optimizing pathways. ARM CPU's are used heavily in consumer electronic devices. This includes smartphones, tablets, and other mobile devices. The reason for usage in these small devices is because int the reduction of the instruction set, this allows for fewer transistors. ARM CPU's are less complex and require less power to operate making them optimal for portable devices.

AARCH64 processor is a variant of ARM. Most features ARM bares, AARCH64 does also. It is also known as ARMv8 and ARM64. This CPU is considered to be new and it is still being developed fully. ARM and its variant AARCH64 processors are supported by Gem5. The reason behind this is because Gem5 was developed to be compatible with the ARM architecture [3]. While Gem5 is in full system mode, it is able to boot with one or multiple processor Linux and very minimal applications that are built with ARM's compilers. Gem5 developers also provides support with Gem5 specific Linux kernels with custom configurations and drivers.

2.2 x86_64

x86_64 processor is the 64-bit version of the x86 instruction set. This processor allows for a karger amount of virtual and physical memory than the 32bit version processor. With x85_64, programs are allowed to store a larger amount of data in memory. This processor provides 64bit general purpose registers and a number of other different enhancements.mx86-64 processors has the capability of booting in a fully backward compatible legacy mode. This is done without 64bit support.

According to Gem5's webpage [1]. X86 support within the Gem5 simulator includes a generic x86 CPU with 64bit extensions, more similar to AMD's version of the architecture than Intel's but not strictly like either. Unmodified versions of the Linux kernel can be booted in UP and SMP configurations, and patches are available for speeding up boot. SSE and 3dnow are implemented, but the majority of x87 floating point is not. Most effort has been focused on 64bit mode, but compatibility mode and legacy modes have some support as well. Real mode works enough to bootstrap an AP, but hasn't been extensively tested. The features of the architecture that are exercised by Linux and standard Linux binaries are implemented and should work, but other areas may not. 64 and 32 bit Linux binaries are supported in syscall emulation mode. We decided to use uclibc library and gcc version 4.9.4 for compiling. After conducting research on which ulib was better, we came to the conclusion that ulibc would allow for a faster compile time (due it its smaller size).

2.3 Benchmarks

Several benchmarks were successfully ran on one architecture. The ARM architecture was the only process we were able to compile and run benchmarks on. We conducted three successful benchmarks provided by the Splash2x suite: LU, a decompose NxN matrix this benchmark is a high-performance computing benchmark, we chose to run this benchmark twice, once with dual core and once with quad core see figure 3. The second benchmark ran was Radiosity this benchmark is used as the use room/large room model. see figure 4.

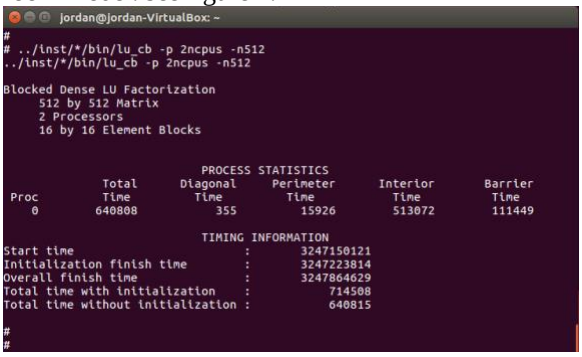

Figure 3, LU_CB benchmark output.

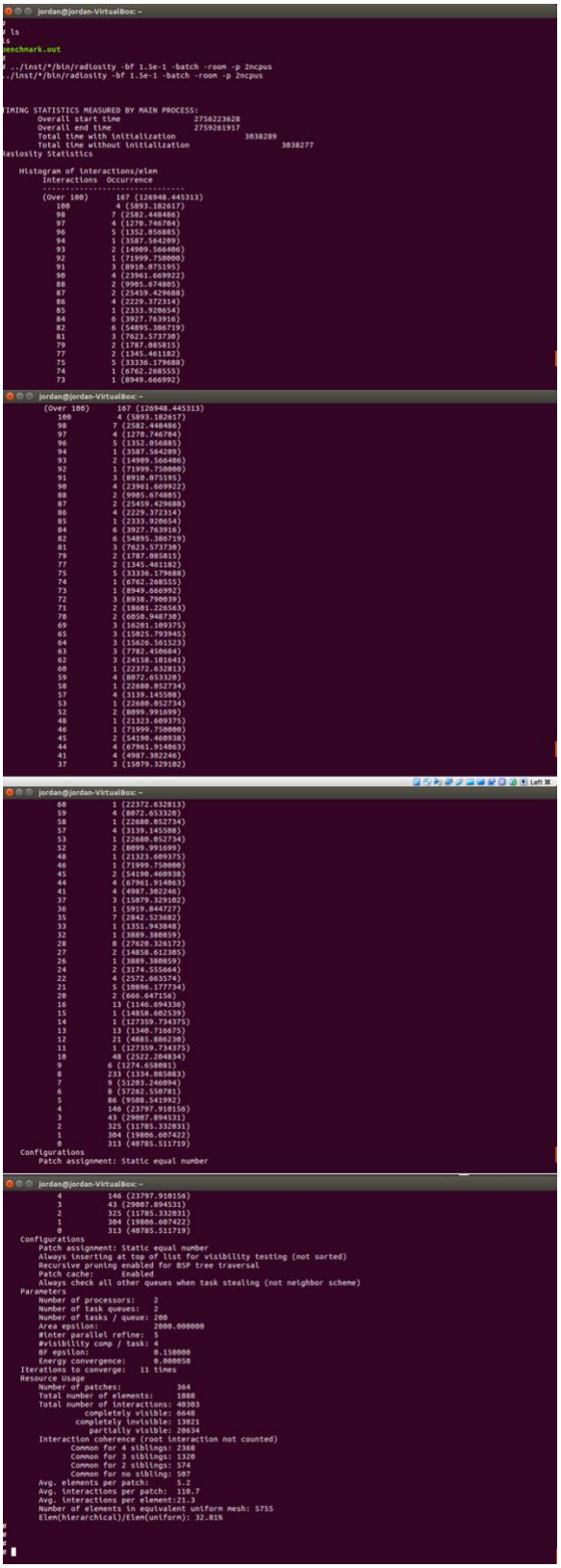

Figure 4, Radiosity benchmark output.

In both benchmarks we conducted, we were able to utilize multi-core. Radiosity benchmark was conducted with 2 cores. Lu_cb benchmark was first ran with 2 cores then a second run with 4 cores.

3 Experimentation

In the experiment we conducted we ran the Splashx2 benchmark twice as stated previously. This benchmark calculates the time it takes to decompose NxN matrix. This experiment we will be decomposing a 512 x 512 matrix with 16 x 16 element blocks. See figure 5 for timing results. We can see the total time to decompose this matrix for both duel and quad core systems are about the same. This is the same case for diagonal decomposition. Perimeter decompose time is almost equal, but duel core takes slightly more time.

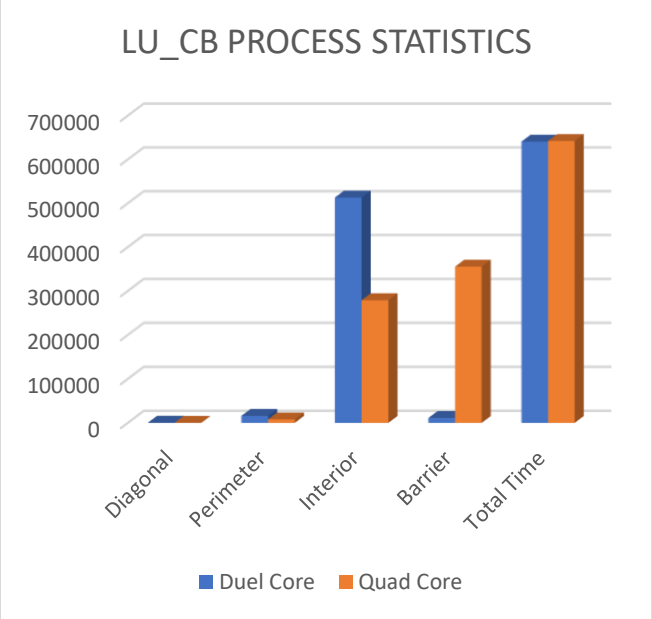

Figure 5, lu_cb benchmark matrix time results.

Interior decompose time there is a huge time difference. The quad-core system outperforms the duel-core by 234433 micro seconds. There a broad gap in time when the barrier is decomposed. Duel core takes a considerably less amount of time of perform the instruction.

4 Related Work

There are several different architecture simulators available besides Gem5. Sniper, Multi2sim and PTLsim [9] are a few that are similar to Gem5.
Sniper simulator Sniper [10] is a parallel simulator for simulating large scale multicore systems using interval simulation. Interval simulation provides a balance between detailed cycle-level simulation and one- IPC

simulation model. Sniper is based on Graphite [11] that supports various one IPC models.

Multi2Sim [12] is another recently developed simulator for CPU-GPU computing, which supports various ISAs. Multi2Sim uses three different simulation models: functional, detailed and event-driven. The detailed simulator and the event-driven module collectively perform timing simulation. Multi2sim supports only OOO execution mode and can support SMT. Memory hierarchy and interconnect networks are highly configurable. Multi2Sim borrows some of its very basic modules from Simplescalar [13]. It does not simulate an OS but still can execute parallel workloads with dynamic threads creation.

The third simulator is PTLsim, a cycle-level fullsystem x86 simulator. It can model a superscalar OOO core with SMT and a simple non-pipelined sequential core which is designed for functional simulation only. The default pipeline model is a modern OOO, based on a combination of features from the Intel Pentium 4, AMD K8, Intel Core 2, IBM Power4/Power5 and Alpha EV8.

Gem5's pipeline configuration flexibility makes it the best simulator in my opinion. PTLsim is also quite flexible in terms of feature configurability. To set the pipeline and other basic configuration parameters some header files have to be changed. PTLsim directly fetches pre-decoded μ-ops from a basic block cache. Branch prediction is highly configurable in PTLsim. It includes a hybrid G-share and bimodal predictors and set-associative BTB, each set to 4k entries. PTLsim supports both distributed and shared issue queues. It is not possible to model a shared instruction queue as well as clusters in PTLsim.

Multi2Sim's configuration is set using two basic configuration scripts: one is used to set memory hierarchy configurations and the other is used to set pipeline configuration parameters. Both are configured according to target configurations. The configured branch predictor is a tournament predictor consisting of a bimodal predictor and two-level adaptive predictor in addition to a set associative BTB with similar sizes as other simulators.

ZSim needs changes in some headers files along with a configuration file to fully model a target architecture. We have changed the header files to add new functional units and configure different pipeline stages to model Haswell pipeline. ZSim does not support a tournament/ hybrid branch predictor. Therefore, a two level branch predictor is modeled. Moreover, ZSim assumes an ideal branch target buffer.

5 Future Work

In future work, we plan to implement x86_64 architecture with gem5. This plan will comprise of several steps. First determining the best kernel for this architecture. Next would be trying a different ulib. We didn't invest much time in trying to compile x86_64. More time would benefit this research also. In this research we ran into conflicting information as to which kernel to use. Better research and trial & error will provide the best solution in this case. Working on the AARCH64 implementation would benefit its ability to run either parsec or splash2 benchmarks. We were unsuccessful in compiling either benchmark. Due to time constraints, we didn't spend much time troubleshooting the reason behind this issue.

Conclusion

In conclusion, buildroot is highly compatibility with Gem5. Buildroots automated system cuts the time to create a bootable image considerably. In the research conducted, we were able to build 2 separate OS images to boot on Gem5, AARCH64 and ARM with a failed attempt to compile x86_64. Two benchmarks were ran on with the ARM processor, Lu_cb and Radiosity. Lu_cb once with duel core and once with quad core. This proves buildroot's highly configurable, menu configuration driven, systems flexibility with creating OS's.

References

[1] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al., "The gem5 Simulator," SIGARCH Comp. Arch. News, vol. 39, pp. 1–7, May 2011.

[2] http://buildroot.uclibc.org/

[3] https://developer.arm.com/research/research-enablement/system-modeling

[4] https://developer.arm.com/research/research-enablement/system-modeling

[5] C.Bienia,S.Kumar,J.P.Singh,andK.Li.ThePARSECBenchmark Suite: Characterization and Architectural Implications. In Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, October 2008.

[6] S.C.Woo,M.Ohara,E.Torrie,J.P.Singh,andA.Gupta.The SPLASH-2 Programs: Characterization and Methodological Considerations. In Proceedings of the 22nd International Symposium on Computer Architecture, pages 24–36, June 1995.

[7] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Es- pasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A Many-Core x86 Architecture for Visual Computing. ACM Trans. Graph., 27(3):1–15, Aug. 2008.

[8] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on, pp. 1–12, IEEE, 2013.

[9] M. T. Yourst, "PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator," in IEEE Int. Symp. on Perf. Analysis of Systems & Software, pp. 23–34, 25-27 April, 2007.

[10] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation," in ACM Int. Conf. for High Performance Comp., Net., Storage and Analysis, pp. 1 – 12, 2011.

[11] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A Distributed Parallel Simulator for Multicores," in IEEE Int. Symp. on High Perf. Comp. Arch., pp. 1–12, Jan. 2010.

[12] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, “Multi2sim: A simulation framework to evaluate multicore-multithreaded processors,” in Int. Symp. on Comp. Arch. and High Perf. Comp., pp. 62–68, Oct. 2007.

[13] T. Austin, E. Larson, and D. Ernst, “SimpleScalar: An Infrastructure for Computer Sys- tem Modeling,” Computer, vol. 35, p. 59.