VerilogTown: Cars, Crashes and Hardware Design

Lindsay Grace
American University
Washington, DC, USA
Grace@american.edu

Peter Jamieson
Miami University
Oxford, OH, USA
Jamiespa@miamioh.edu

Naoki Mizuno
Miami University
Oxford, OH, USA
e-mail address

Boyu Zhang
Miami University
Oxford, OH, USA
e-mail address

ABSTRACT
VerilogTown is a game about cars, crashes and hardware design. The game is designed to help teach and practice the hardware description language, Verilog. The game uses the metaphor of traffic signals to help players understand and practice the code needed to implement combinational and sequential logic in digital circuits. Borrowing from the emerging space of human computation games, player solutions in the games can be directly transferred to embedded system for real world use.

Author Keywords
Games and learning; synthesizable circuits; FPGA; ASIC; Verilog; hardware description languages.

ACM Classification Keywords
H.5.m. Information interfaces and presentation

OVERVIEW
Understanding the complex logic needed to code Verilog can be challenging for many students. Much of the research in employing the power of games to teach code has focused on traditional software programming. While the benefits of such work have been discussed, there is little being done for the even more complex space of hardware design. As physical computing interests [1] increase and as the demand for well trained engineers continue [5], the demand for tools to help engage learners in hardware are expected to increase[8].

VerilogTown seeks to support this need by providing a very simple set of tools for practicing both combinational and sequential logic in hardware description languages. While many software developers are familiar with object oriented and procedural development models, fewer are familiar with the circuit theory behind combinational and sequential logic. These circuit theory concepts are essential to developing solutions in Verilog. Solutions encoded in Verilog can be use in FPGAs (Field Programmable Gate Arrays), ASICs (Application Specific Integrated Chips), or other digital/analog silicon technologies including creating a processor (a.k.a. a computer).

Combinational logic is continuously processed. It is time independent logic without the ability to store state. The common analogy for combinational logic is water where water is never stopped and always flows.

Sequential logic, on the other hand is controlled by a clock which serves as a metronome of the circuit. In sequential logic, the present state of the logic input and the sequence of inputs are a factor. Where combinational logic is a stateless flow of water, sequential logic is a flowing water stopped by a dam then released on every tick of the metronome. In the most base terms, sequential logic is combinational logic with state.

VerilogTown shown in figure 1, takes this analogy and adapts it to the flow of traffic. Players must design logic solutions to prevent the flow of traffic from causing crashes. Players do so by managing traffic signals. To test

Figure 1. Verilogtown screenshot, depicting gameplay with varied land and water vehicles.

ACE '15, November 16-19, 2015, Iskandar, Malaysia
ACM 978-1-4503-3852-3/15/11.
http://dx.doi.org/10.1145/2832932.2832936

their solution, players write Verilog and watch the resulting traffic flow. Their goal is to get all of the vehicles through traffic without a crash. Players can then optimize their solutions, creating more efficient solutions while playing.

The game includes a Verilog editor, templates, and 10 logic puzzles. The idea is to wrap the challenge of learning Verilog with the pleasures of a playful experience. From the literature [2] it is clear that such efforts should encourage experimentation, learner persistence and focus. The game is implemented in Java and may be played on any Windows, Linux, or Mac OS machine.

This game takes as its motivation the previous successes of environments like Scratch [7] and LOGO [3]. It extends that precedent by applying the real world applicability of human computation games [6] to this classroom tool.

GAME PLAY

Players are presented with one simple goal – get the vehicles on the map to their destination without crashing. Much like real world traffic management, some solutions are better than others. If players choose to hold traffic too long, the logic is inefficient. If players choose to move traffic poorly, there will be crashes. Each level contains an ever increasing set of challenges circuit code. Players can choose to implement a combinational or sequential logic solution for each of the problems.

To play, players choose an intersection map of varying complexity. The simplest maps have but a few traffic signals and limited traffic, the most complex have up to 10 intersections to manage. Figure 2 demonstrates the gameplay phases and complexity.

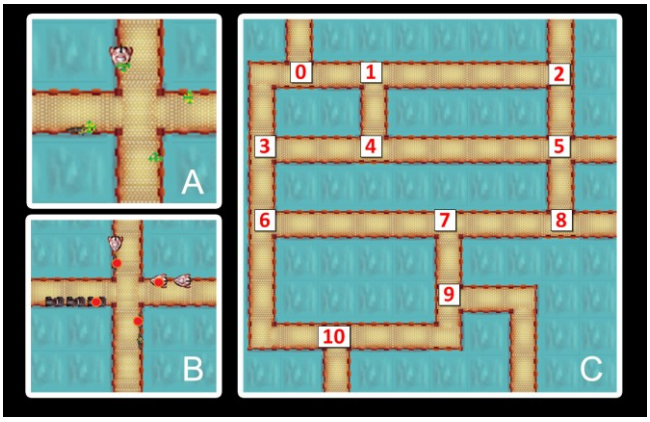

Figure 2. Screenshots from the game. Section A demonstrates a simple, single car intersection. In B, multiple cars come to an intersection with the possibility of collision. C demonstrates a complex 10 intersection puzzle.

The player must design and implement logic for each of the traffic signals in their map. A simple double click launches the embedded Verilog editor as shown in figure 3. Players can use the embedded editor which provides typical editing functions (e.g. undo, execute, save) or they can paste their solutions as text from another editor.

Once a set of logic is completely designed, players simply return to the simulation window to watch their results unfold. Cars start moving, traffic jams occure or traffic flows freely based on the code behind each traffic managing module the player has created.

Once complete, the players are presented with basic information about the number of crashes (which they can witness), number of vehicles that have completed their journey succsesfully, and the amount of time the solutions took to execute as shown in figure 4.

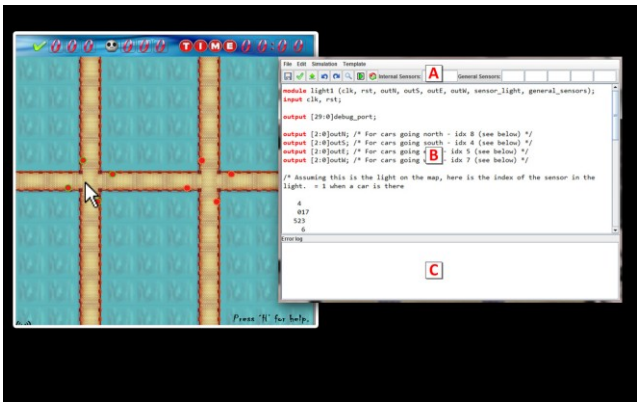

Figure 3. The embedded Verilog editor. Section A is the toolset for editing Verilog. Section B is the code input and editing window. Section C is for error reporting.

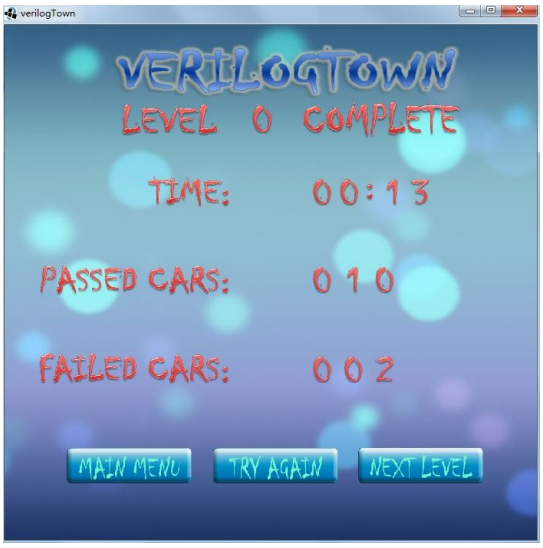

Figure 4. Screenshots of player results

The game was designed to allow players to embrace failure and practice without the frustration and anxiety common to more serious applications. The game provides the typical accoutrements expected of a casual game. While the game is running, players can pause the traffic simulation to understand what has gone right or wrong via the "P" key. They can also seek functional help at any time via a simple

"H" key press. A retry option is also readily available for retesting a solution to determine where it has failed or succeeded. The complete diagram of available game actions and results are shown in figure 5.

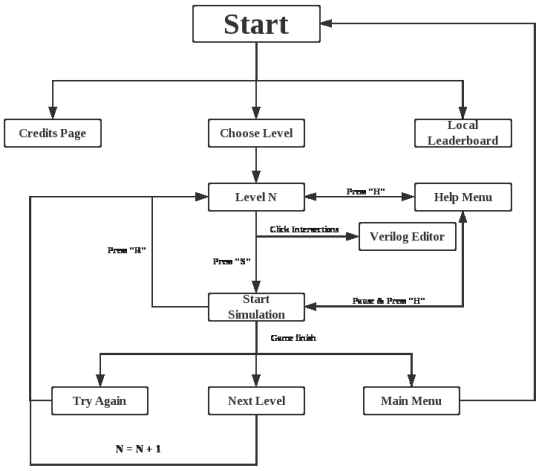

Figure 5: Game process flow diagram

The benefits of framing hardware programming in this way are twofold. First, players are allowed to witness the errors or successes in their logic visually. While traditional environments might simply demonstrate logic errors and require users to evaluate why they are happening, the game environment of Verilogtown is in itself a visualization of the results of logic control. This bridges the obvious gap between intended result and resulting scenario. Players see what went wrong creating a more satisfying feedback loop than traditional programming environments.

Secondly, as mentioned, the code generated within the game is true Verilog. Players need only cut and paste the code they generate within Verilogtown's editor for use in Verilog. This means that while people play and learn within the Verilogtown environment, their code can be easily transferred for use. This is an extension of human computation play, as levels can be designed to address specific logic sets for which multiple players can prescribe varied Verilog solutions.

CONCLUSION

Individuals who are particularly fond of programming may argue that all coding is a game, in much the same way that individuals who are fond of any activity (e.g. art, automobile maintenance, bookkeeping, etc) can find play in it. The benefit of designed experiences like Verilogtown is in their ability to bridge this play gap. The game employs the cognitive task of programming a hardware description language through a game. In short it facilitates the interpretation of programming challenges as play by employing functional Verilog to solve in-game problems. As such the game represents an early foray in to combining the benefits of human computation play to resolving non-game problems. As mentioned, Verilog developed to solve problems in Verilogtown can simply be cut and paste for real world use. Ultimately the core goal of the project is pedagogic – aiming to facilitate an interest in practicing the unique logic challenges in programmable hardware using Verilog.

The game is available at no cost to engineering faculty and students for us in their classrooms by following the url http://www.users.miamioh.edu/jamiespa/verilogTown/. The game has been in development for over a year and has been used as a successful practice tool in Miami University second year undergraduate course. The researchers are now making the game available to a wider audience to support education.

ACKNOWLEDGMENTS

The game, VerilogTown, was created as institutional collaboration between Miami University and the American University Game Lab. The researchers would like to thank the partnering institutions for their support.

REFERENCES

1. Blikstein, P. 2013. Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In Proceedings of the 12th International Conference on Interaction Design and Children (pp. 173-182). ACM.
2. Brown, S. L. 2009. Play: How it shapes the brain, opens the imagination, and invigorates the soul. Penguin.
3. Feurzeig, W. 1969. Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project.
4. Hayes, B., & Games, I. 2008. Making computer games and design thinking: A review of current software and strategies. Games and Culture.
5. Langdon, D., McKittrick, G., Beede, D., Khan, B., & Doms, M. 2011. STEM: Good Jobs Now and for the Future. ESA Issue Brief# 03-11. US Department of Commerce.
6. Quinn, A. J., & Bederson, B. B. 2011. Human computation: a survey and taxonomy of a growing field. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 1403-1412). ACM.
7. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. 2009. Scratch: programming for all. Communications of the ACM, 52(11), 60-67.
8. Jamieson, Peter (2010). Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat?. Proc. of Intl. Conf. on Frontiers in Education (FECS), 289-294.