VerilogTown: Cars, Crashes and Hardware Design

Peter Jamieson, Miami University
Lindsay Grace, American University Game Lab
Naoki Mizuno, Miami University
Boyu Zhang, Miami University
Josh Collins, Miami University
Alex Williams, Miami University
John-Rhys Garcia, DePaul University

Abstract: *VerilogTown* is a game about cars, crashes and hardware design. The game allows players to learn, practice and play with the Verilog Hardware Description Language (HDL). The game asks players to solve a variety of traffic puzzles using digital logic designs specified through an HDL, which is what hardware designers use to create applications synthesizable to integrated chips. This paper outlines the design and final product, including the fundamental benefits of such approach. It is provided as a case study in domain specific game design that should prove useful to other researchers looking to employ the potential of play to facilitate learning of complex systems, models and theories.

Introduction

One of the core foundations in continued support of STEM educational research is an appreciation for understanding logic. Logic pervades a variety of disciplines and practices ranging from writing to mathematics, from hypothesis to synthesized solutions. It is no surprise that games have provided an appropriate solution for practicing logic. Traditional games, like Chess, have been championed as appropriate ways to practice logic and envision multivariate outcomes (Ferguson, 1995).

The core benefit of such play are also outlined by Brown and others (2009). Psychology and allied disciplines have demonstrated that a playful mindset supports flexible problem solving and a focused state (Brown, 2009). The research findings, state simply—we are at our best when we are at play. One challenge is that many traditional games, like Chess, rely on a single model of logic. Such games, like many non-digital games, are sequential games, where player logic is the sequenced response to another player's decision. It is also difficult to transfer the direct experience of solving logic problems in such games to the practical challenges of non-game contexts.

Digital games clearly extend the possibilities of education and training, particularly through practicing logic. It is clear that the future of learning is likely to involve game solutions, whether viewed from the last decade or from a contemporary perspective (Shaffer et al, 2005). The foundation of computer science certainly supports the execution and practice of sequential logic, but it also affords for other approaches. Digital play extends the traditional bounds of game logic by affording for experiential and multivariate logic. Players of digital games can test and retest logical hypothesis. They can also isolate and refine their logic through simulation. Examples of this include everything from squad management in *Call of Duty* to urban planning in *Civilization*, or balancing social and psychological needs in *The Sims*. The potential and demonstrated benefit of such games based learning has been demonstrated by Kurt Squire (2005) and others in domains as varied as psychology (Tannahill, Tissington, & Senior, 2012) to creative writing (Gerber and Price, 2011).

Games also include more exotic examples that change, through epistemological effect, the nature of knowing and how logic is executed. Imperfect knowledge, for example is the cornerstone of such games as Battleship, which require players to logically deduce location but obscures those locations physically (by placing a panel between players). Practicing logic through imperfect knowledge is a structure common to both games (via obscured data) and educational assessment (e.g. fill-in the blank or argumentation contexts).

Digital games take this epistemology further, by offering perfect knowledge and creating play from a perfect knowledge scenario. Beyond providing omniscient views of complex situations, as many aerial perspective games do, digital games afford for increasingly complex opportunities. Jonathan Blow's Oracle Billiards (2003) is one such example. The game allows players to see where billiard balls will land before the player hits them with the cue ball. In such a design, the nature of billiards is fundamentally changed. The game matriculates from simple physics simulation, to an exercise in longer term strategic planning. What the digital game affords is a series of long term decisions normally only available to the most experienced professional billiards player. Players aren't worried about one ball, they are worried about the result of two or three steps after sinking a target.

In this history of blending the benefits of games and education, several examples illustrate its potential. Of particular relevance is the LOGO (Feurzeig 1969) language system, which coupled a simple programming language with the ability to express creative computer graphics. LOGO inspired a generation of young learners to see the logical and even mathematical relationship of rendered graphics and code. Modern compliments like Scratch (Resnick et al 2009) offer similar appeal.

While much of the discussion about teaching logic has focused on the merits of learning software programming this research offers a solution in hardware programming. The constraints of hardware programming have often made such work less than accessible. However, the obvious opportunity in Arduino (Jamieson, 2010) and similarly intentioned solutions has made the opportunity more alluring. Likewise, the continued use of LOGO technology (Weinberg and Yu, 2003) indicates the value of such experiential education. These are commercial, software programmable, hardware solutions. They are playful opportunities to learn relatively complex logic systems. They allow people to problem solve by playing with their solutions.

Recent work in human computation games has inspired a variety of researchers and game designers to harness the problem solving in these games to address real world issues. In short, human computation games (HCG) not only allow people to practice problem solving in games, they record and apply player solutions to real world problems. Games such as *FoldIt* have resounded in the academic research community as offering an entirely new paradigm to employing play to solve a myriad of non-game problems (Cooper et al, 2010). This has resulted in an explosion of such games of which the most common activities are labeling data (Grace and Jamieson, 2013).

The researchers set forth to tap the potential of playful learning and human computation games by developing a game. The game, *VerilogTown* employs the Verilog hardware description language to create synthesizable logic designs. The solution our team of researchers created is a distinct combination of these two approaches. It brings the power of playful learning to the problem solving success of human computation games. The researchers developed *VerilogTown*, as an experiment in making synthesizable logic more accessible. Players must manage the flow of traffic in the virtual world, to avoid accidents and traffic jams. The solutions they develop in-game can be used for digital hardware solutions outside the game.

The game's problem, managing traffic, is one that is commonly understood. It also serves as an analogy for the solutions the players are generating. By employing, synthesizable Verilog HDL to solve problems in the game, players are producing real synthesizable circuits. Put simply, these circuits can be configured on a device and deployed in the field. The means of circuit design is via a hardware description language (HDL) created by designers, which is a more efficient way of designing circuits compared to schematic entry.

This research was conducted as a collaboration between subject matter experts in their respective fields. It includes a well-respected researcher in FPGA, or field programmable gate array, one of several programmable integrated circuit technologies. *VerilogTown* is also the product of efforts from a well-respected academic game designer, several artists, and computer engineering researchers. The game serves as an appropriate case study for highlighting specialized, playful learning. By sharing this research, we hope to make such work more accessible and encourage others to apply similarly designed solutions to their specific domains.

Game Overview

VerilogTown is a game about cars, crashes and hardware design. The goal of *VerilogTown* is to get the cars safely through the streets to their intended destination. As shown in figure 1, cars drive toward intersections that are managed by traffic lights. Players create rules for the traffic lights, which manage the intersections. The rules, are articulated in Verilog to control each of the traffic lights in a level's puzzle. Puzzles range from simple, single intersection challenges, to multi-intersection, multi car calculations.

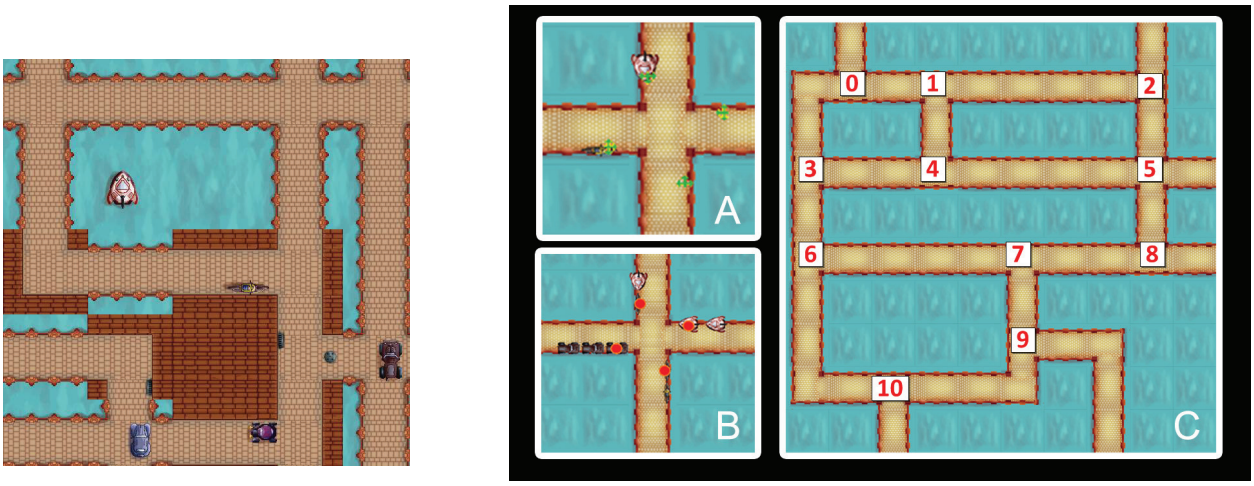

Figure 1: Screenshots from the game. Section A demonstrates a simple, single car intersection. In B, multiple cars come to an intersection with the possibility of collision. C demonstrates a complex 10 intersection puzzle.

Much like a typical tower defense game, players cannot control the non-player characters. Instead players must design logical solutions that balance the fastest solution with avoiding a collision. Players, could for example, choose to hold all the cars at a single intersection for a long period of time. However, much like conventional traffic management, doing so may result in a longer than necessary waiting time for everyone. Likewise, changing a signal while cars are still in intersections can also create problems. Players must edit the traffic light logic (including directional management) to optimize solutions. Figure two demonstrates how editing a signal works.

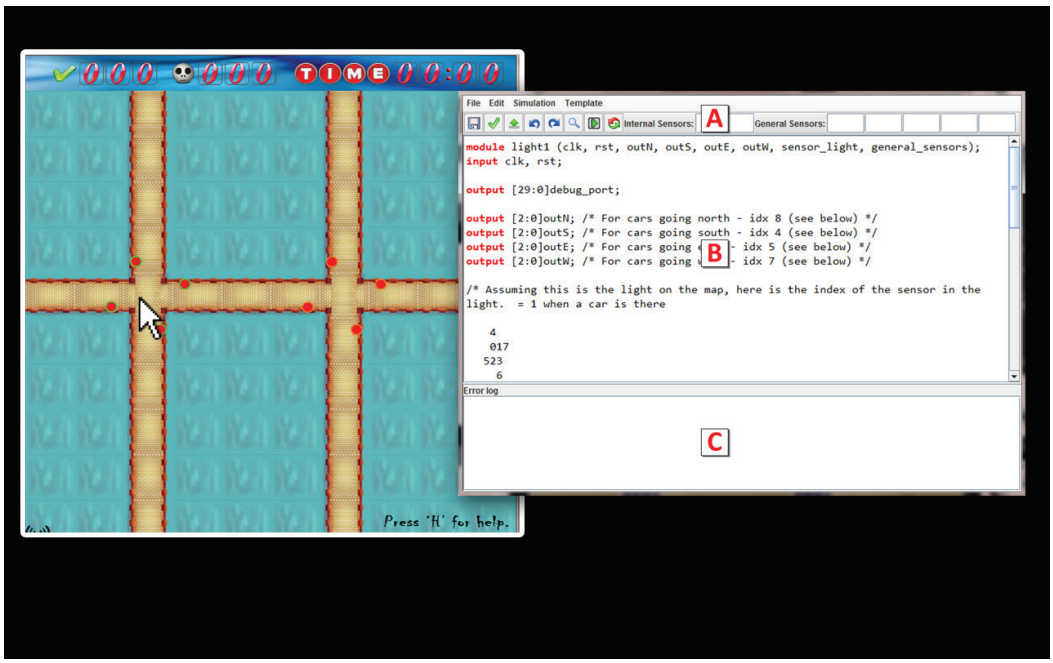

Figure 2: Players edit logic by selecting a signal at an intersection and adjusting the Verilog code in the editing window.

Embracing the research literature on play, players learn by failing. The game is designed for players to run through the scenario, see the complex multivariate scenario and then experiment for a solution. Each iteration through executing logic and seeing the result drives toward an optimal solution.

The game also provides players the ability to create their own puzzles to solve or map to existing challenges they have in synthesizable circuits. The ability for players to create their own puzzles affords for a virtual playground supporting understanding, experimentation and practice of the relatively complex logic employed in digital circuit

design. This is one of the hallmarks of the approach. Instead of merely subscribing to a set of prescribed challenges, players are supported by player generated content and self-initiated challenges. This simple design decision did complicate development, but it ultimately provides for a far more robust, scalable and engaging game.

The game is provided as a free download on the website[1]. The game can be played on any Microsoft Windows machine supporting Java. The interface is minimal and supports key problem solving actions. These include zooming in and out, panning through the level map, starting pausing, restarting, and requesting basic help files. On the website players can also provide their solutions and learn about the programming structure.

Game Design Highlights

The game is designed to take advantage of three key characteristics. These are the, the clarity of analogy, simultaneous visual representation, and playful learning. The analogy of cars moving through traffic maps nicely to circuit routing. Without knowing the science of circuits, players have a clear sense of their goal, the multiple factors impeding their goal and how to meet their goals. They must simply manage the gates, or traffic signals, that manipulate flow.

The visual representation of this process provides a second layer of learning support. Circuits employ multiple activities at once. Asking players to envision such activity in their minds is both taxing and distracting. Thinking through states with only a mental model works for low complexity tasks, but it is immediately complicated by the addition events or additional gates. Seeing a representation of what is happening aids in understanding the process, the multiple factors, and the solutions required.

Lastly, the analogy and visualization are designed to support a play state. Instead of reprimanding players with error messages and negative feedback, the players experience crashes. More importantly, they are supported in one of the hallmarks of play's value—experimentation. In the *VerilogTown* environment players are encouraged to try and learn from their mistakes. It is very much founded on the principle that play is an educational playground (Brown, 2009).

Because of the nature of circuit design, *VerilogTown* serves as a playful way to learn two relatively complex notions in logic design. The game supports both combination and sequential circuit logic.

Practicing Complex Logic: Combinational and Sequential

The games content focus is in the playful execution of digital circuit theory. There are two important models in digital circuit theory, Combinational and Sequential logic. The former is a subset of the later and can be considered divergent ways of understanding logic design for digital circuits. The game supports both. Unlike the following paragraphs that explain such logic, players are afforded experiential learning. They can play through these two approaches, fail gracefully within the game system and continue to experience the difference in these logic approaches.

Combinational logic is continuously processed. It is a time independent logic that has no means for storing state. On analogy is that Combinational logic is a waterway where the water is never stopped and always flows from start to end.

By analogy, sequential logic is controlled by a clock which serves as the heartbeat of a circuit. In sequential logic, the present state of the logic input and the sequence of inputs are a factor.

To adapt the water analogy, it is helpful to think of sequential logic as water flowing. That water is stopped by a dam and then released every time a second ticks. When this second ticks, we would have a 1Hz clock frequency for the circuit. In its simplest, sequential logic can be considered as combinational logic with state. The state is the dam, which can store for a limited amount of time. The concept is somewhat similar to timer based software operations common to games, computer graphics and monitoring systems as sequential logic is the basis for constructing finite state machines. Fine State Machines (FSM) are commonly employed by computer software.

Combinational and sequential logic can be mixed together in a sequential circuit. This is normal. In reality, any sequential logic circuit will have combinational elements. However, a purely combinational circuit has no sequential logic.

VerilogTown provides two templates for players to use in understanding, practicing and playing with the difference between combinational and sequential logic. The first template is a combinational template. It demonstrates checking an in-game traffic signal and executing turn one direction to Go if there is no car in the intersection and

there is a car waiting. The second Verilog template is a sequential template that changes the light every 3 seconds in a counter-clockwise pattern. Using these two templates, players can experiment with how to solve the gradually more complex puzzles in the game. Figure 3 provides an excerpt of the Verilog logic, demonstrating the core programming logic required.

```
begin
  if (rst == 1'b0)
  begin
    count <= 8'd0;
    start <= 3'd0;
    outN <= Go;
    outS <= Stop;
    outE <= Stop;
    outW <= Stop;
  end
  else
  begin
    count <= count + 1'b1;

    if (start < 3'b111)
    begin
      outE <= Go;
      outN <= Stop;
      if (count == 8'd255)
        start <= start + 1'b1;
    end
    else
    begin
    case (count)
      8'd0 :
      begin
        outN <= Stop;
        outE <= Go;
      end
      8'd64:
      begin
        outE <= Stop;
        outS <= Go;
      end
      8'd128:
      begin
        outS <= Stop;
        outW <= Go;
      end
      8'd192:
      begin
        outW <= Stop;
        outN <= Go;
      end
    endcase
    end
  end
end
```

Figure 3: Verilog code excerpt used to solve an in game puzzle.

As expected, players can cut and paste their *VerilogTown* solutions into other tools for use on synthesizable circuits. In short, they can play with an in-game solution until they solve their non-game needs.

Conclusion

The goal of this brief case study was simply to illustrate the designed solutions for incorporating domain specific logic into a playful game experience. The research team sought to move beyond the feared, chocolate covered broccoli solution. We did not want to merely cover the challenging task of programming synthesizable circuits with sloppily applied set of game-like sweetness. Instead, we aimed to apply the contemporary theory in human computation games with the overarching mantras of engaging educational game design. The game aims to make this complex task accessible by embracing the hallmarks of playful learning and experiential education. Players are encouraged to experiment, supported in their failure, and provided an experience that maps directly to real world application. The hope is while this specific application is not widely understood, this case study provides design reference for game makers seeking to address similar challenges in narrow, but essential domains.

References

Brown, S. L. (2009). Play: How it shapes the brain, opens the imagination, and invigorates the soul. Penguin.

Blow, Jonathan (2003). Oracle Billiards. Available at http://number-none.com/blow/prototypes/ Last played, 1/15/2015.

Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., ... & Popović, Z. (2010). Predicting protein structures with a multiplayer online game. Nature, 466(7307), 756-760.

Ferguson, R. (1995, January). Chess in education research summary. In Chess in Education A Wise Move Conference at the Borough of Manhattan Community College.

Feurzeig, W. (1969). Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project.

Gerber, H. R., & Price, D. P. (2011). Twenty-first-century adolescents, writing, and new media: Meeting the challenge with game controllers and laptops. English Journal, 68-73.

Grace, L. D., & Jamieson, P. Gaming with Purpose: Heuristic Understanding of Ubiquitous Game Development and Design for Human Computation. Handbook of Digital Games, 645-666.

Hayes, B., & Games, I. (2008). Making computer games and design thinking: A review of current software and strategies. Games and Culture.

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.

Shaffer, D. W., Halverson, R., Squire, K. R., & Gee, J. P. (2005). Video Games and the Future of Learning. WCER Working Paper No. 2005-4. Wisconsin Center for Education Research (NJ1).

Squire, K. (2005). Changing the game: What happens when video games enter the classroom. Innovate: Journal of online education, 1(6).

Tannahill, N., Tissington, P., & Senior, C. (2012). Video games and higher education: what can “Call of Duty” teach our students?. Frontiers in psychology, 3.

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.

Weinberg, J. B., & Yu, X. (2003). Robotics in education: Low-cost platforms for teaching integrated systems. Robotics & Automation Magazine, IEEE, 10(2), 4-6.

Acknowledgments

The game, *VerilogTown*, was created as an institutional collaboration between American University and Miami University. The researchers would like to thank the American University Game Lab for their support in disseminating this research.

1 http://www.users.miamioh.edu/jamiespa/verilogTown/