Mario Breakout

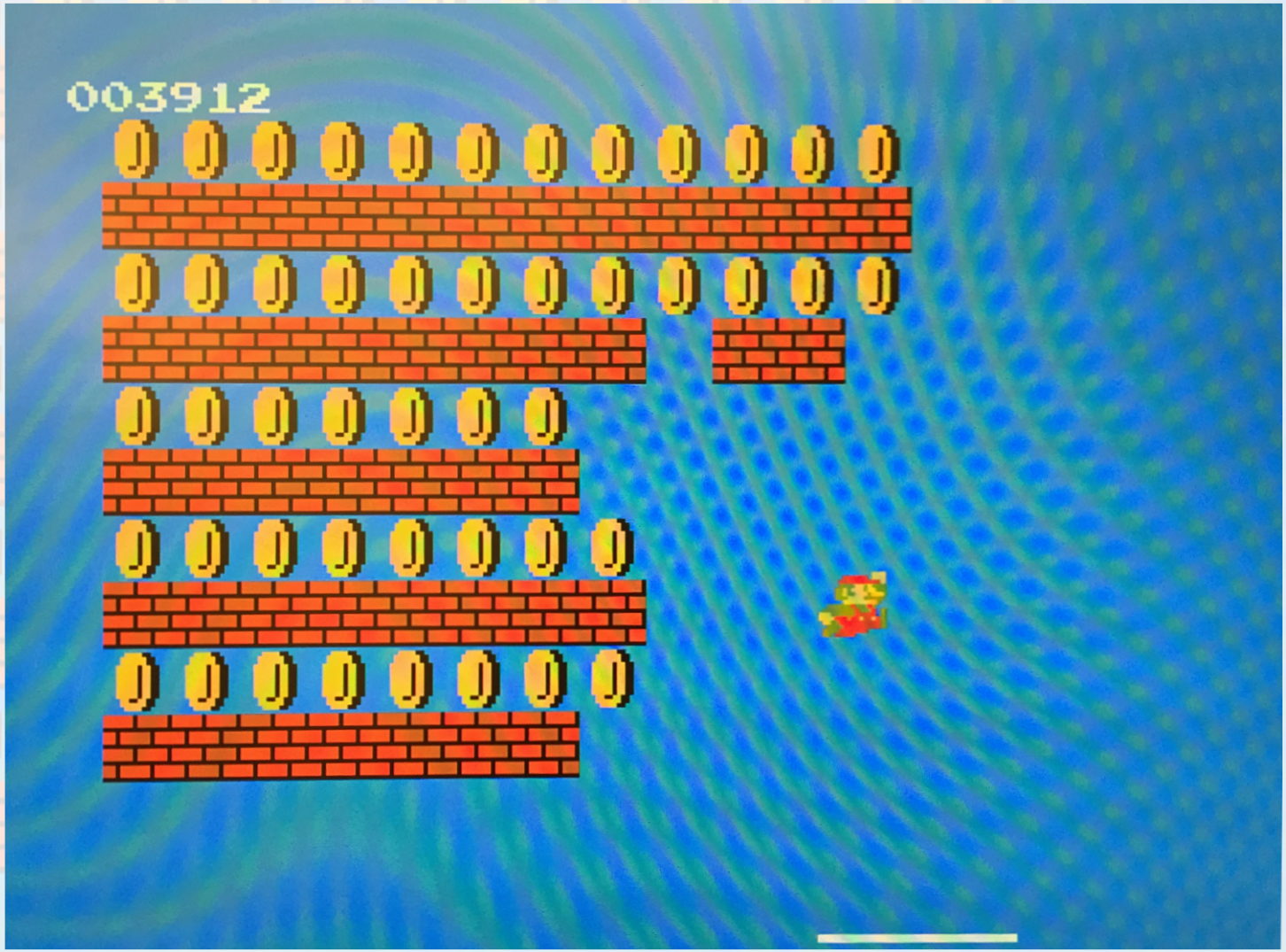
Adam Dalentoft, dat11ada@student.lu.se
Simon Wallström, dat11swa@student.lu.se
Viktor Sannum, dat11vsa@student.lu.se

Design of Embedded Systems Advanced Course

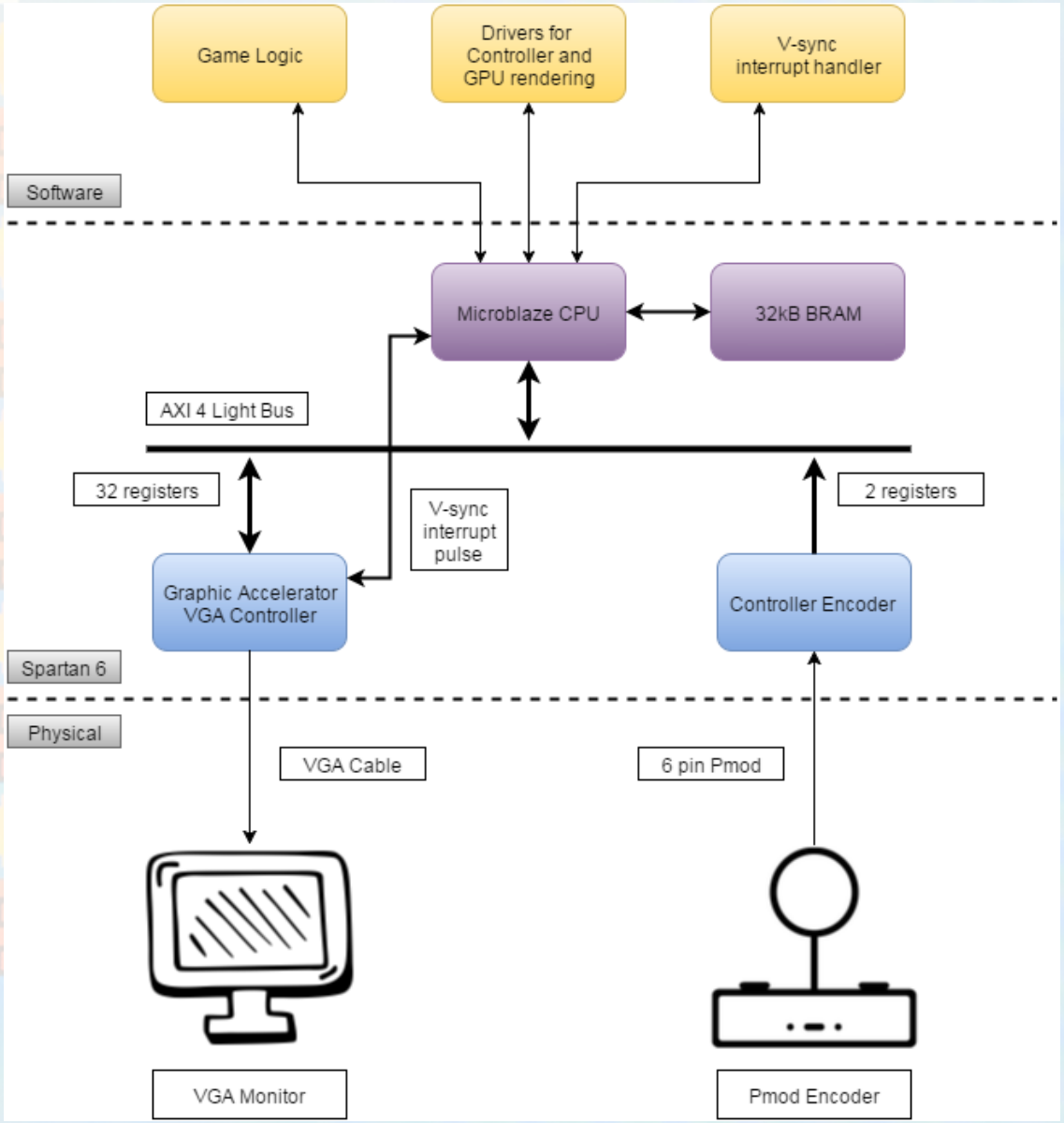
October 15, 2015

Summery

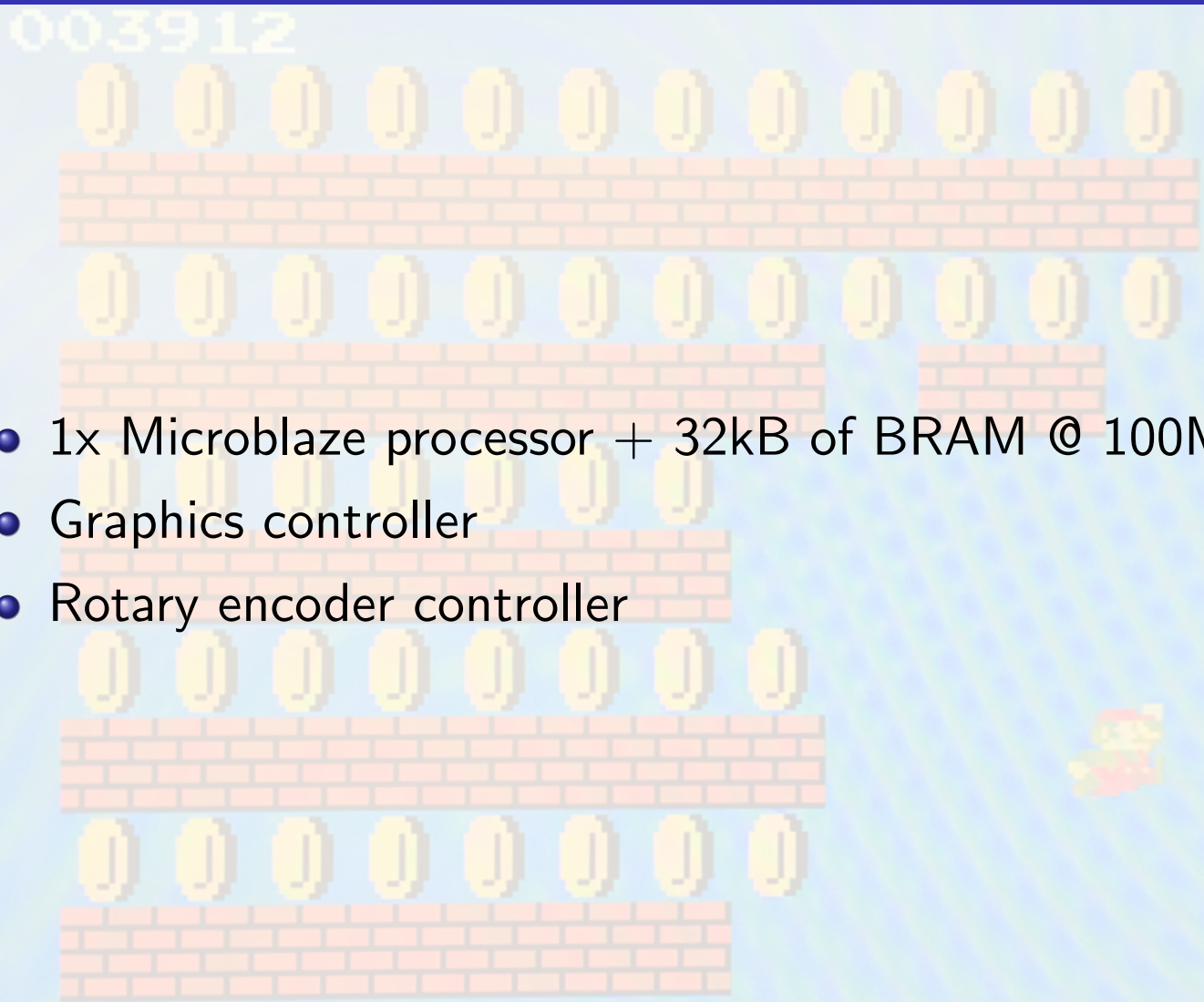
Introduction - Mario Breakout



Introduction - Architecture



Overview - Hardware

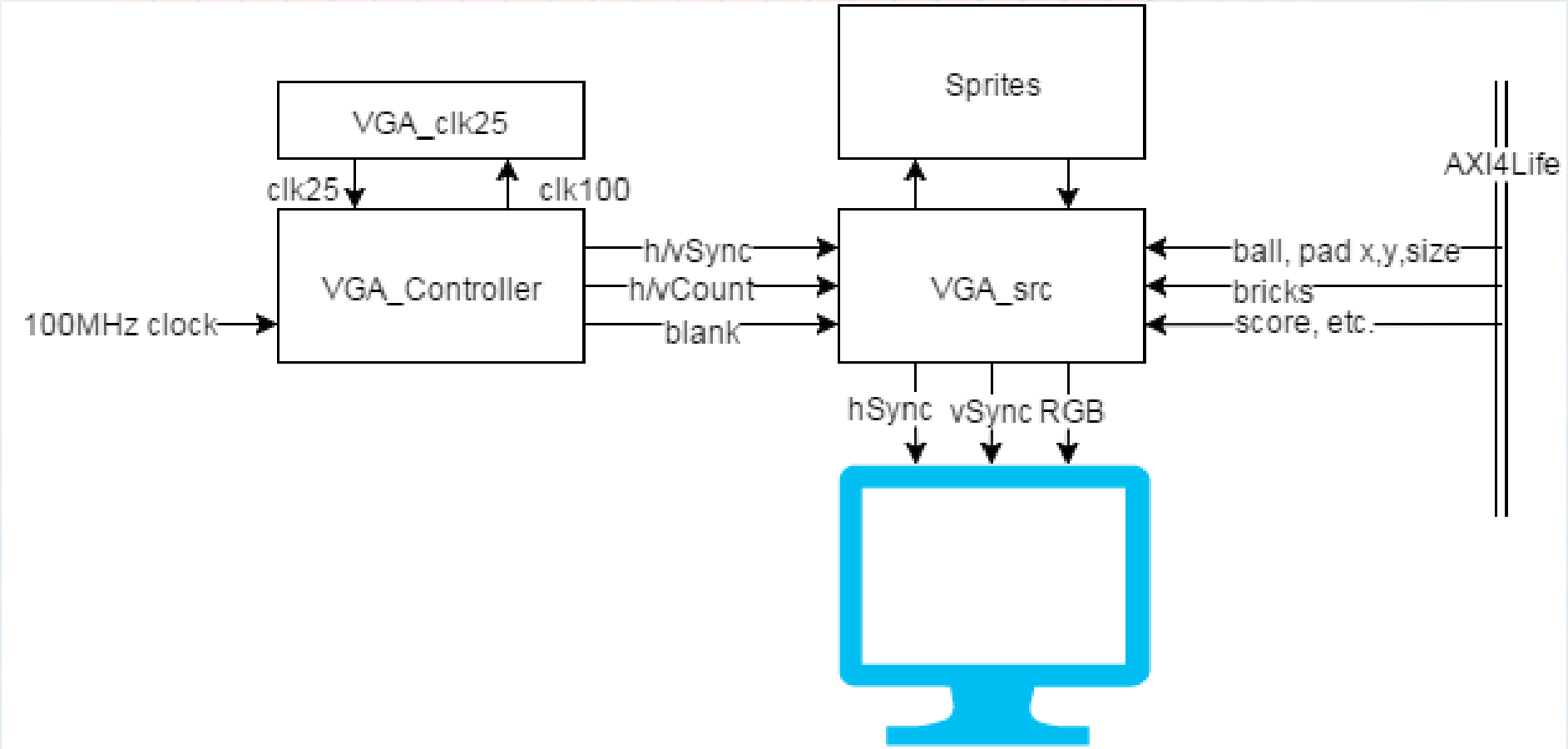


- 1x Microblaze processor + 32kB of BRAM @ 100MHz
- Graphics controller
- Rotary encoder controller

Graphics

- VGA clock controller
  - Generates a 640x480@60Hz VGA signal
- Hardcoded sprites (arrays of std_logic_vectors...)
- Sprite drawing logic
- 32 axi4lite registers holding game data from software

Graphics - architecture



Controller

- One dimensional controller, move the paddle left or right
- First implementation with on board push buttons
- Final implementation with Rotary Encoder using Axi4Lite registers

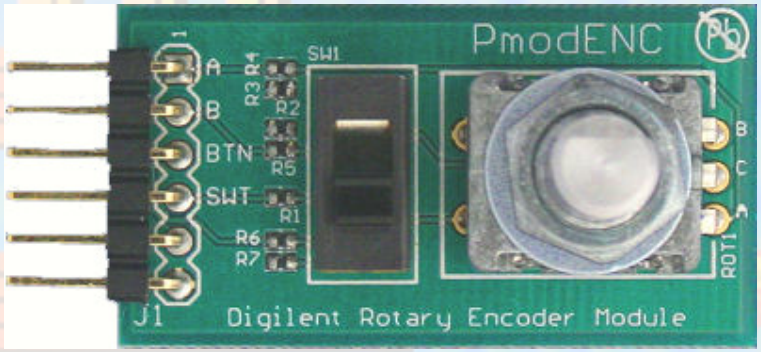
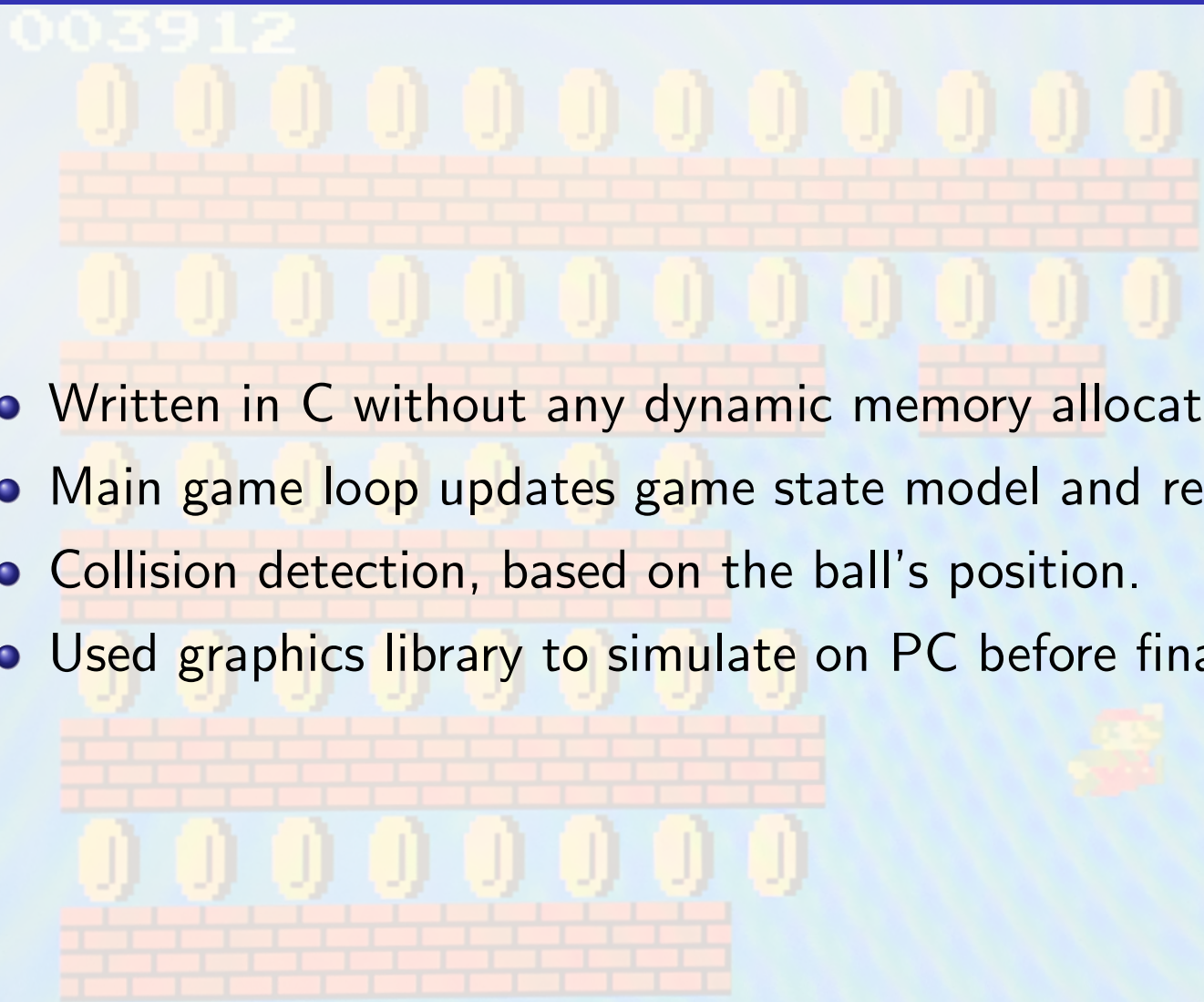


Figure: Diligent Pmod Rotary Encoder, used as controller for the paddle[1].

Hardware/Software interface

- AXI4Lite bus register for communication between components
- Processor reads from knob controller, writes to graphics controller registers
- Interrupt handler in sw receives interrupt on vSync which iterates the game loop
- AXI logic in hardware is auto generated, only allows $\leq$32 32-bit registers, could be improved.

Game logic

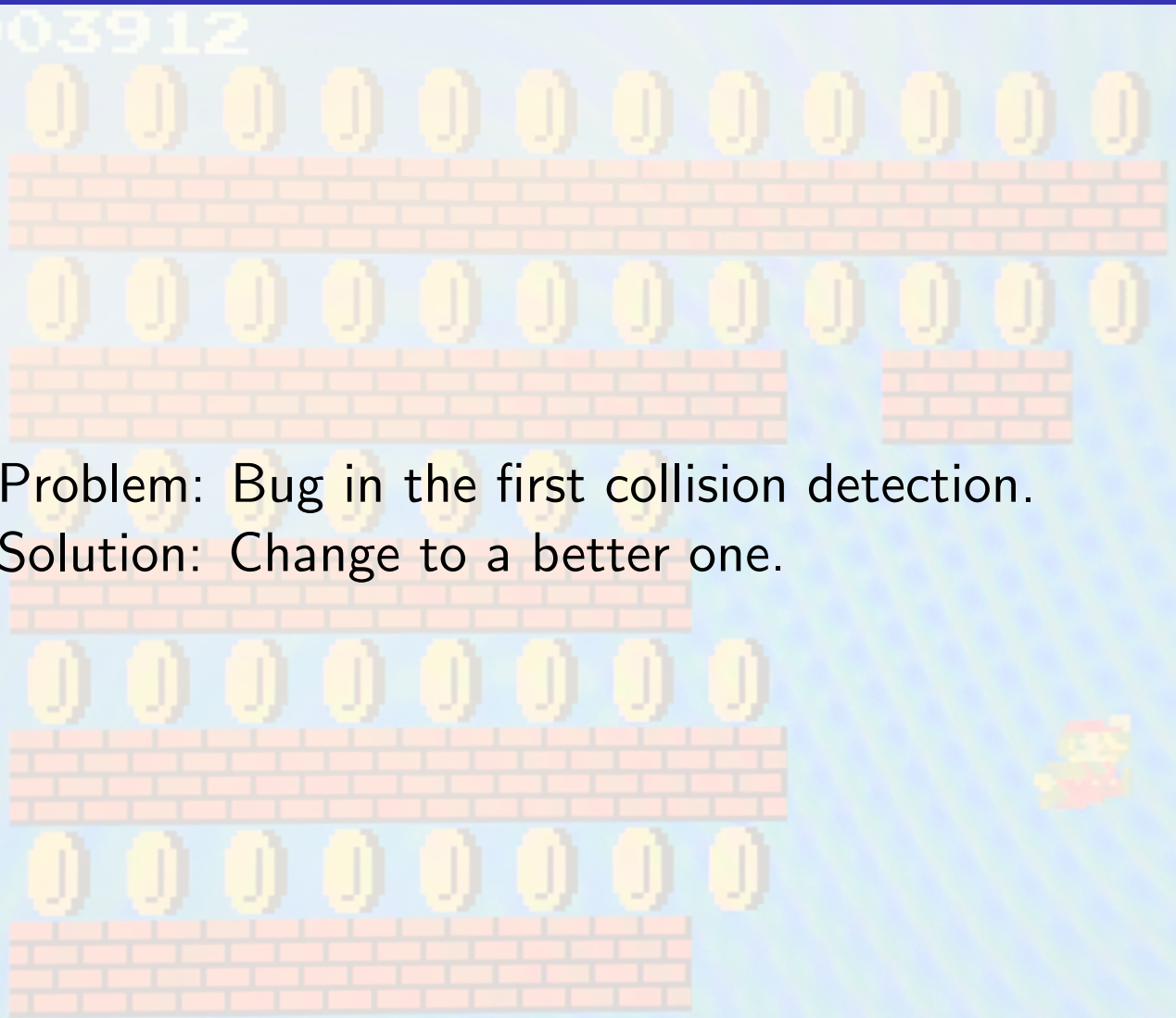


- Written in C without any dynamic memory allocation.
- Main game loop updates game state model and redraws any changes
- Collision detection, based on the ball's position.
- Used graphics library to simulate on PC before final implementation.

Hardware problems and solutions

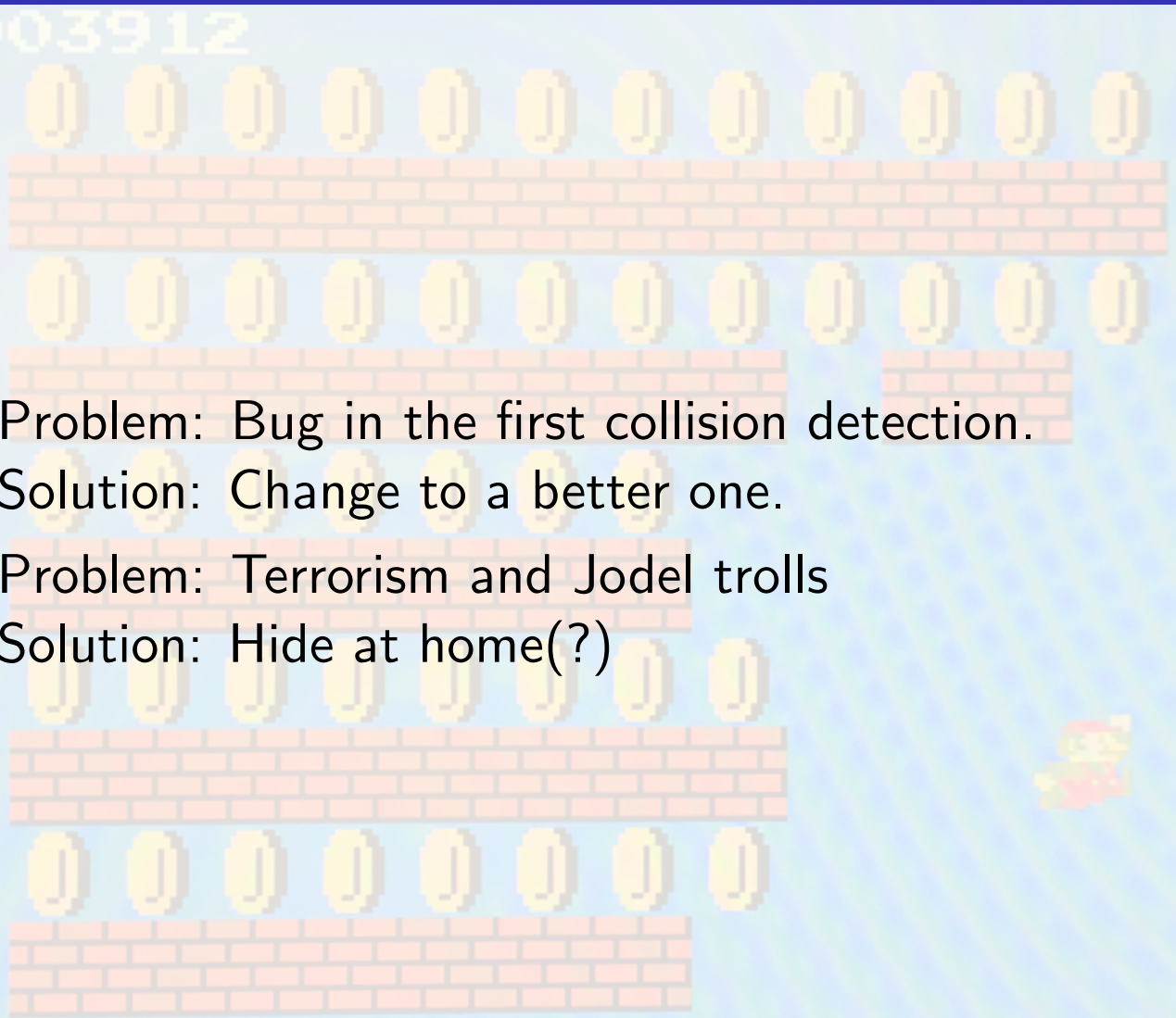
- Problem: First version of VGA controller did not work.
  Solution: Changed to a other solution.
- Problem: Hard to debug hardware.
  Solution: Using test benches on some modules.
- Problem: Noise on the rotary encoder outputs.
  Solution: Filter the output before using the data.

Software problems and solutions



- Problem: Bug in the first collision detection.
  Solution: Change to a better one.

Software problems and solutions



- Problem: Bug in the first collision detection.
  Solution: Change to a better one.
- Problem: Terrorism and Jodel trolls
  Solution: Hide at home(?)

Possible improvements

- Highscore.
- Sound and music.
- More score and life handling.
- More brick types.
- More efficient sprite handling logic.
- Nicer knob.
- Particle effects.

Lessons learned

- Think through the implementation before compiling it as the compilation takes time in hardware.
- Don't get stuck on a track if it takes too long time, try to find another solution.
- Find out which tasks are time consuming before starting implementation to divide the group in a better way.

Questions

Thanks for listening!

Questions?

References

Diligent store site for 'PmodENC - Rotary encoder'. http://digilentinc.com/Products/Detail.cfm?NavPath=2,401,479&Prod=PMOD-ENC (28/9-15)