CSEE 4840 · EMBEDDED SYSTEMS · SPRING 2026

MAZE CHASE

A hardware-accelerated pursuit game on the DE1-SoC

Cyclone V FPGA · ARM HPS · Linux · VGA · USB HID

Ryan Lo (yl5972) · Junhao Qu (jq2434) · Boxiong Li (bl3155) · Kevin Liu (kl3755)

Prof. Stephen A. Edwards · Columbia University

What is Maze Chase?

A 30 × 20 grid pursuit-evasion game.

Player (evader) moves with USB-keyboard arrows.

Two **ghosts** chase via BFS on the maze graph.

To win: stand on each of 6 special tiles for 10s.

All 6 charged → gate opens at the top.

Reach the gate before a ghost catches you.

640 × 480

VGA · 60 Hz

30 × 20

tile grid · 16 px each

16 regs

Avalon-MM · 32 b each

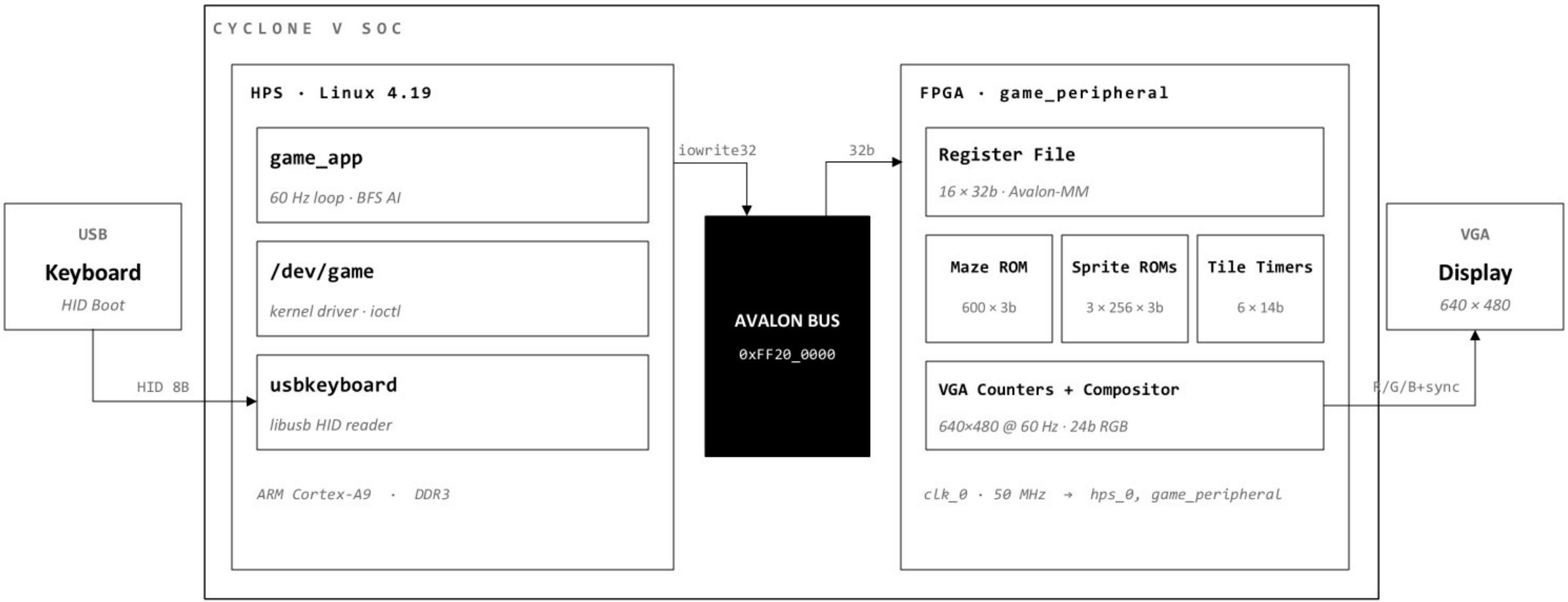
5 Hz

movement tick · 12 frames

DATA FLOW

USB keyboard → game_app.c → /dev/game (ioctl) → Avalon-MM @ 0xFF200000 → game_peripheral → VGA

System Architecture



Register Map · Avalon-MM @ 0xFF20_0000

16 registers · 32 bits each · 4-bit word address · 64 bytes total

Offset	Word	Name	Dir	Bit fields
0x00	0	**EVADER_POS**	R/W	[9:5] row · [4:0] col
0x04	1	**GHOST0_POS**	R/W	[9:5] row · [4:0] col
0x08	2	**GHOST1_POS**	R/W	[9:5] row · [4:0] col
0x0C	3	**TILE_STATUS**	R	[5:0] = 6 "activated" flags
0x10–0x24	4–9	**TILE0..5_TIME**	R	[13:0] frame count · 0..600 (10 s)
0x28	10	**GAME_STATUS**	R/W	[3]=reset · [2]=gh_win · [1]=ev_win · [0]=gate_open
0x2C	11	**FRAME_SYNC**	R	[16]=vsync_tick · [15:0]=frame#
0x30–0x3C	12–15	**reserved**	–	read as zero

POSITION ENCODING

writedata[9:0] = {row[4:0], col[4:0]}. Five bits each is enough for a 30 × 20 grid.

05

PART 1 · HARDWARE

FPGA Game Peripheral

game_peripheral.sv · VGA pipeline · ROMs

06

Inside game_peripheral.sv

REGISTER FILE · sources of game state, written by HPS

evader_pos · ghost0_pos · ghost1_pos · game_status · 6 tile_timers · frame_count

vsync_tick from VGA counters drives the 6 tile_timer instances (saturating counter, activated when count ≥ 600 frames = 10 s)

vga_counters

640×480 @ 60 Hz
hcount/vcount
VGA_HS/VS/BLANK

→

coord math

pixel→tile (col,row)
maze_addr = r·30+c
sprite_addr = (y,x)

→

ROMs (M10K)

maze_rom · 600×3b
sprite ROMs · 256×3b
.hex from gen_mif.py

→

compositor

priority mux
maze < tiles < ghost < evader
palette → 24b RGB

→

VGA out

VGA_R/G/B[7:0]
VGA_CLK = hcount[0]
VGA_BLANK_n

VSYNC_CDC · 2-FF synchronizer + rising-edge detect

vga_counters runs off 50 MHz (clk50). The vsync_pulse used to update tile_timer and frame_count is brought into the system-clock domain via meta/sync/prev FFs, then edge-detected so we get exactly one 1-cycle strobe per frame.

07

ROMs: source images → .hex → on-chip M10K

Source assets

evader - hardcoded
ghost0 - hardcoded
ghost1 - hardcoded

maze layout hard-coded
in gen_mif.py
→ also writes maze_data.h
(shared with HPS)

→

gen_mif.py

Read hardcoded with PIL
Map RGB → 3-bit color
index (8-color palette)

Write *.hex (sprite ROM
$readmemh format)
Write *.mif (maze ROM
Quartus Memory Init)

→

FPGA on-chip ROM

maze_rom 600 × 3 bits
→ M10K block
sprite_rom × 3 each
256 × 3 bits → MLAB/M10K

Read every pixel cycle
by compositor
Total memory: 4,104 bits

08

PART 2 · PLATFORM

System Integration & Build Flow

Qsys · soc_system_top.sv · Makefile

Qsys system · soc_system.qsys

THREE COMPONENTS IN THE SYSTEM

clk_0

Clock Source

50 MHz · feeds every clock input

hps_0

Cyclone V Hard Processor System

Cortex-A9 + DDR3 + I/O. Exports h2f_lw_axi_master.
LWH2F bridge ENABLED, full H2F + F2H disabled.

game_peripheral_0

Custom IP (Maze Chase Game Peripheral, v1.0)

Avalon-MM slave @ 0x0000 in bridge space · size 0x40
clock from clk_0 · vga conduit exported

ADDRESS MAP

HPS view · 32-bit physical address

0xFF20_0000

Lightweight HPS-to-FPGA bridge base

game_peripheral_0 @ offset 0x000

absolute 0xFF20_0000 – 0xFF20_003F (64 B)

0xFF20_0040 – 0xFFFF_FFFF (unused; no IP)

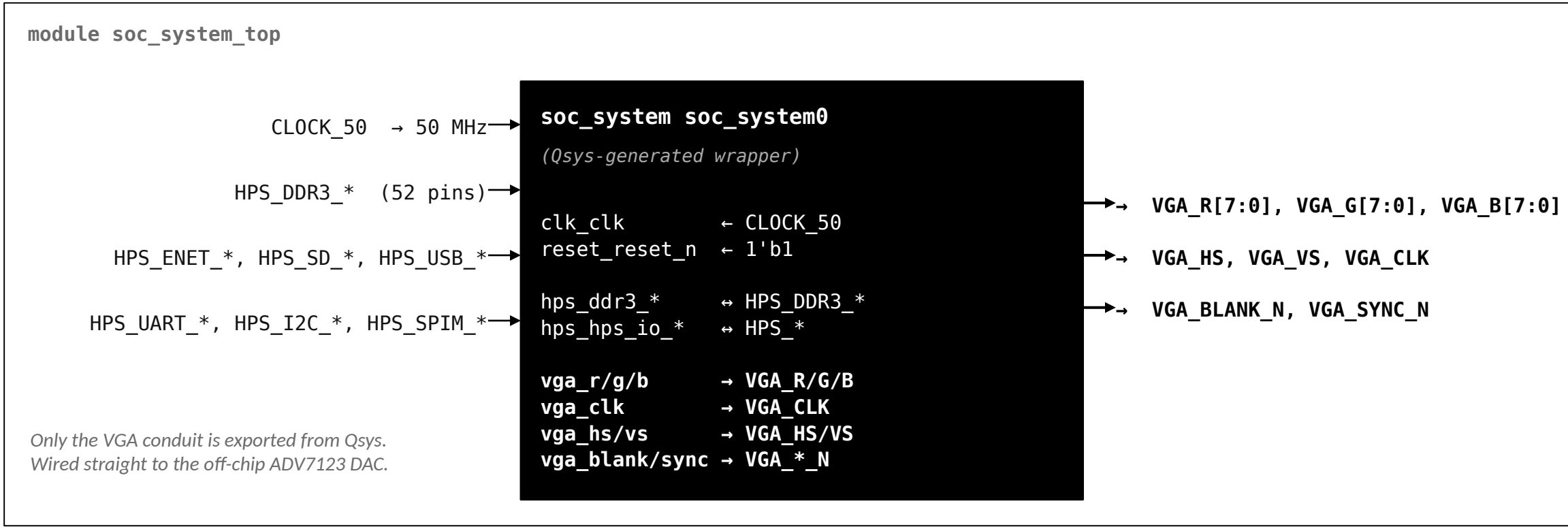
WHY 64 BYTES?

16 regs × 4 B/reg. Address bus is 4 bits (16 words).

DTS NODE GENERATED BY sopc2dts

compatible = "csee4840,game_peripheral-1.0" · reg = <0x1 0x0 0x40> · matched by kernel driver

Top-level wiring · soc_system_top.sv

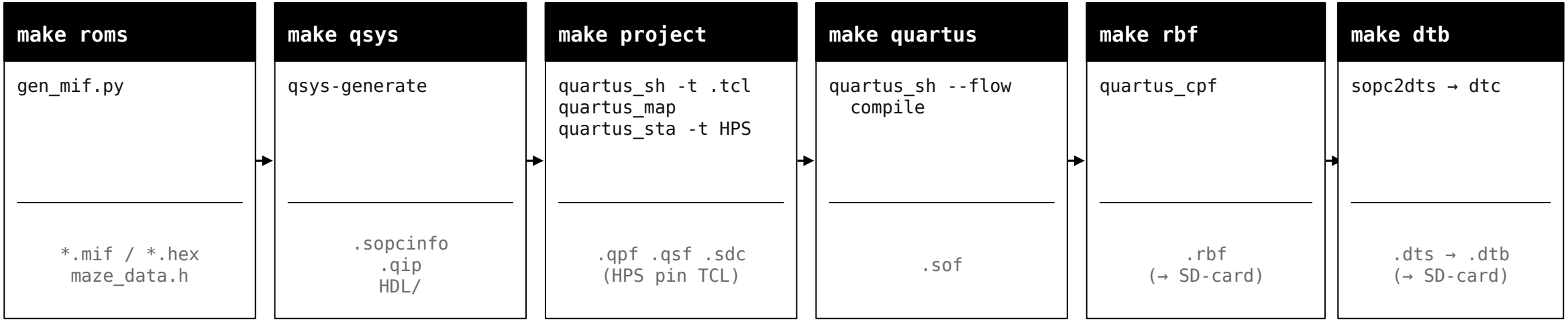


UNUSED-PERIPHERAL SQUELCH

```
assign GPIO_0 = SW[1] ? {36{SW[0]}} : {36{1'bZ}};   // HEX, LED, GPIO, AUD, DRAM, ADC, PS2, TD  →  tied to SW
```

Suppresses Quartus "no driver" warnings on unused board peripherals; identical pattern as lab3 vga_ball.

Build flow · Makefile



MODIFICATIONS OVER lab3-hw/Makefile TEMPLATE

Added: ROM auto-generation

- new ROM_FILES variable (8 files)
- .PHONY: roms → python3 gen_mif.py
- make qsys now depends on $(ROM_FILES)
- rom-clean rule for housekeeping

Removed: ~80 lines (no longer needed)

- preloader / U-Boot / kernel-build targets (reuse the SD-card Linux from lab2)
- associated BSP / KERNEL / CROSS vars
- corresponding *-clean targets

12

PART 3 · SOFTWARE

HPS Game Application

game_app.c · 60 Hz loop · BFS ghost AI

game_app.c · 60 Hz main loop + BFS AI

MAIN LOOP PACING

ioctl GAME_WAIT_VSYNC

blocks until FRAME_SYNC ticks → 60 Hz

read keyboard buffer

shared with USB pthread (mutex)

if (frame % 12 == 0)

advance game state at 5 Hz

evader = step(evader)

validate against maze_data.h

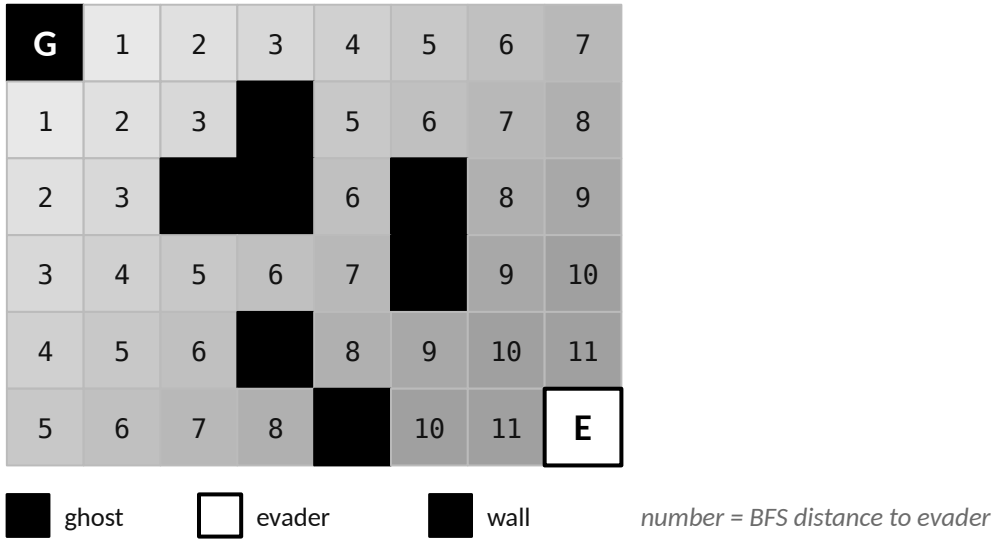
bfs_step(ghost0,1)

next cell on shortest path

ioctl GAME_WRITE_STATE

push positions + status to FPGA

BFS GHOST AI



IMPLEMENTATION

Each ghost step: flood-fill from evader, pick neighbour with min distance.
O(R·C) = O(600) per ghost per tick.

14

PART 4 · SOFTWARE

Kernel Driver & USB Keyboard

game.c (/dev/game) · usbkeyboard.c

Linux device driver + USB keyboard reader

game.c · /dev/game

platform_driver_probe()

matches DTS "csee4840,game_peripheral-1.0".
Driver runs only if its DT node is loaded.

of_address_to_resource → of_iomap

ioremaps 0xFF20_0000 (64 B) into kernel VA.
Doesn't hard-code the address — DT provides it.

misc_register → /dev/game

MISC_DYNAMIC_MINOR; userspace opens /dev/game like any char device.

unlocked_ioctl

GAME_WRITE_STATE · GAME_READ_STATE · GAME_RESET_GAME · GAME_WAIT_VSYNC.
Uses iowrite32 / ioread32 on the BAR.

usbkeyboard.c · libusb HID Boot

libusb_init + device enumeration

Walk USB device list; match class 0x03 (HID), subclass 0x01 (Boot), protocol 0x01 (Keyboard).

libusb_claim_interface

Detach kernel driver if attached (usbhid).
Userspace owns the keyboard endpoint.

libusb_interrupt_transfer

Polls IN endpoint for 8-byte HID Boot report:
[modifier, _, key0..key5].

Why not /dev/input/eventN?

Composite keyboard exposes Boot + Consumer + System subdevices.
Arrow keys only come from Boot.

Resources & timing

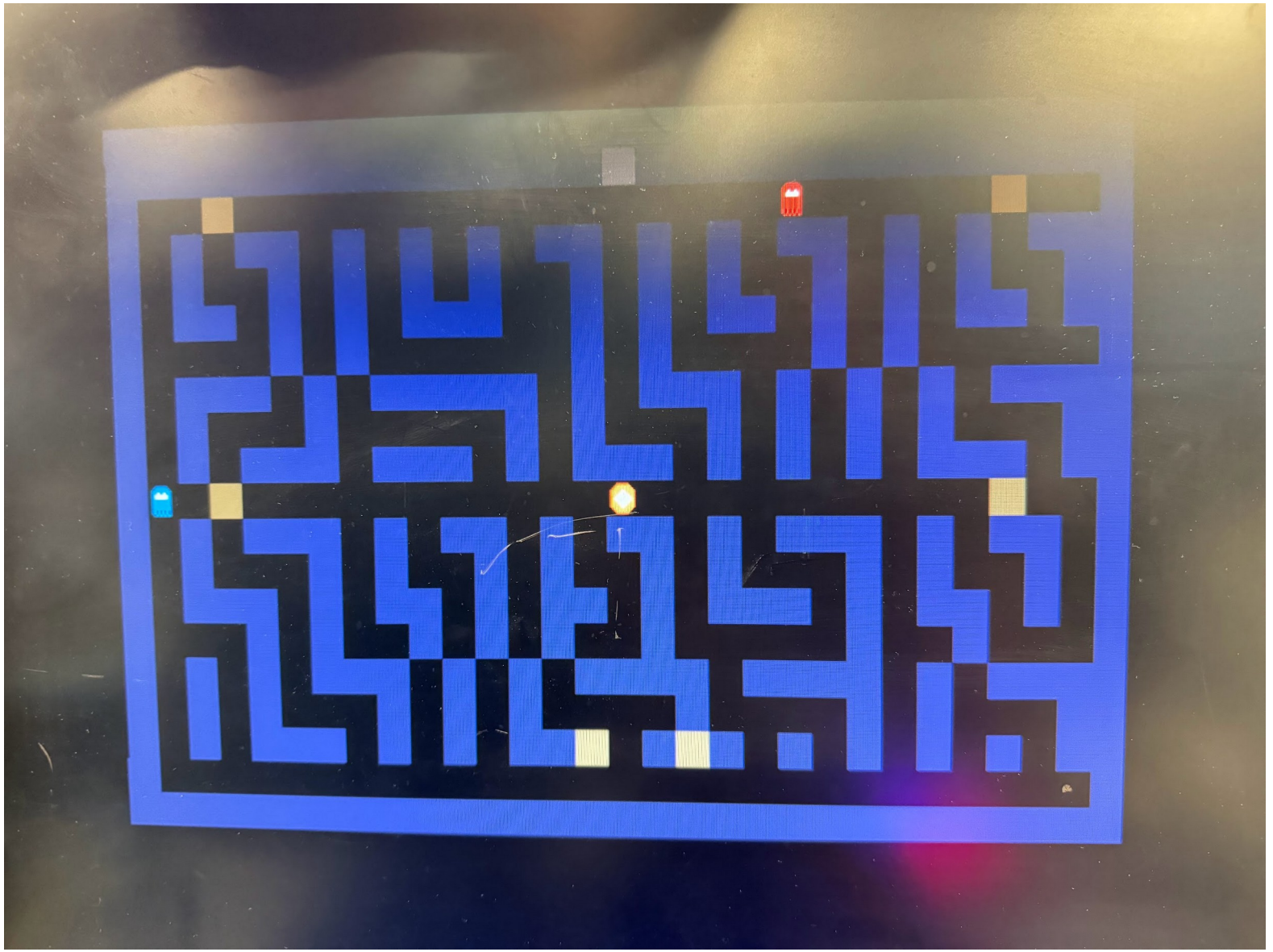
397	2	0	141.62
ALMS NEEDED	**M10K BLOCKS**	**DSP BLOCKS**	**FMAX (MHz)**
1% of 32,070	1% of 397 · 4,104 mem bits	0% of 87	clock_50_1, 1100 mV / 85°C required ≥ 50 MHz ✓

SOURCE CODE WE WROTE

(machine-readable in final_project-hw.tar.gz / final_project-sw.tar.gz)

File	Lang	Lines	Role
game_peripheral.sv	SystemVerilog	567	VGA pipeline · register file · ROMs
soc_system_top.sv	SystemVerilog	283	DE1-SoC top-level wrapper
game_peripheral_hw.tcl + gen_mif.py	Tcl + Python	360	Qsys component + PNG→.hex/.mif
Makefile (hw)	Make	144	qsys-generate → quartus → rbf · dtb
game_app.c	C	446	60 Hz loop, BFS AI, USB thread
game.c + game.h	C (kernel)	325	platform driver, ioctl, /dev/game
usbkeyboard.c/.h + Makefile (sw)	C + Make	173	libusb HID Boot reader + xc compile

LIVE DEMO



Thank You!

CSEE 4840 · Embedded Systems · Spring 2026

Ryan Lo (yl5972) · Junhao Qu (jq2434) · Boxiong Li (bl3155) · Kevin Liu (kl3755)