

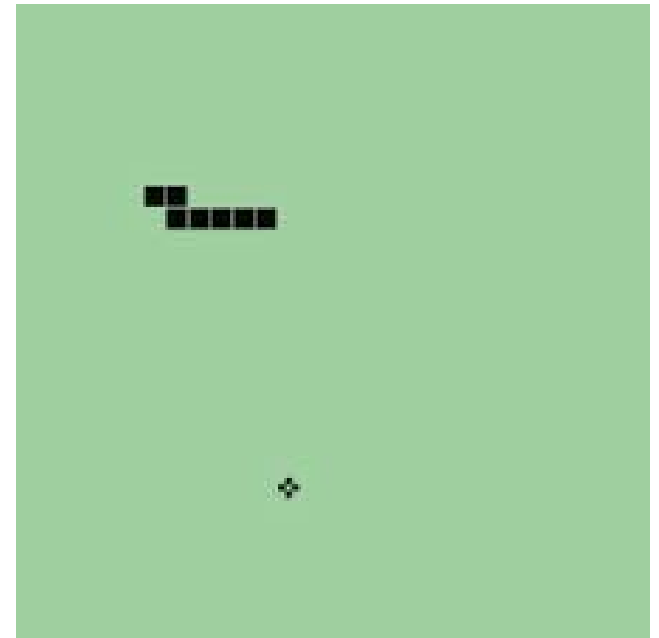
SNAKE+



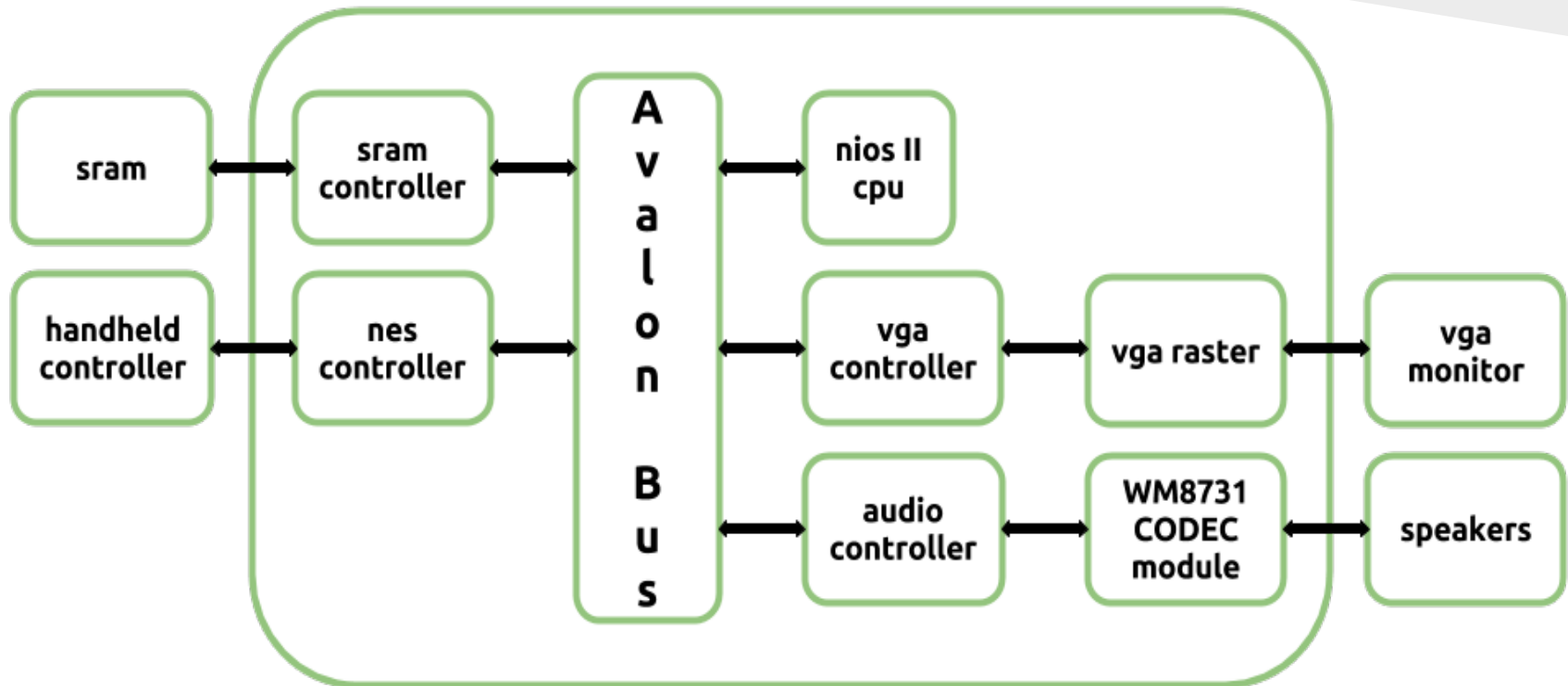
Nina Berg
Joseph Corbisiero
Ilan Elkobi
Molly Karcher
Brian Wagner

Overview and Objectives

- Remember that addictive Nokia game from the '90s?
- Remake it on an FPGA, but do it better:
 - better graphics
 - multiple players
 - nes controllers!
 - power-ups
 - obstacles

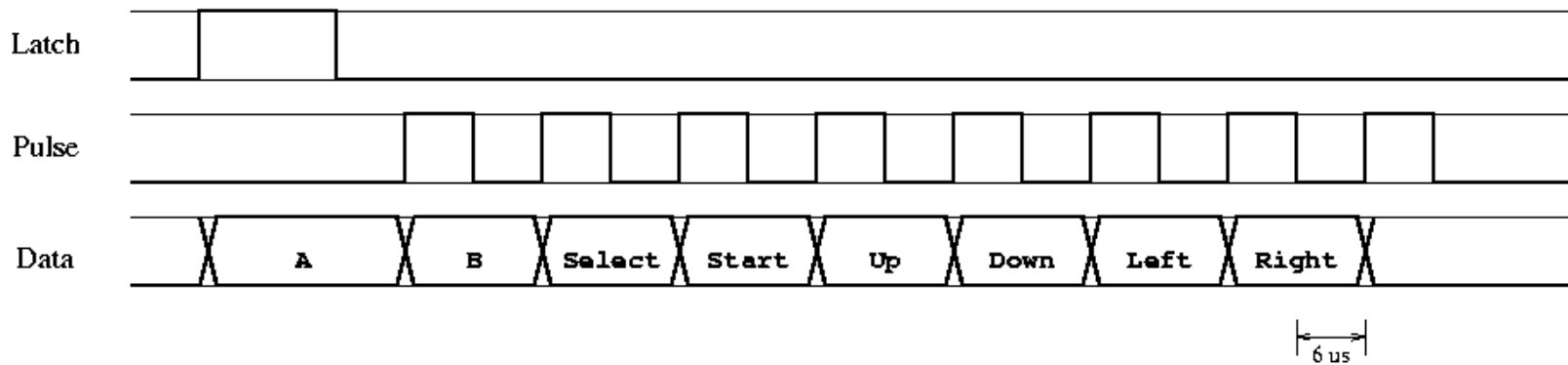


Design Architecture



Design Architecture

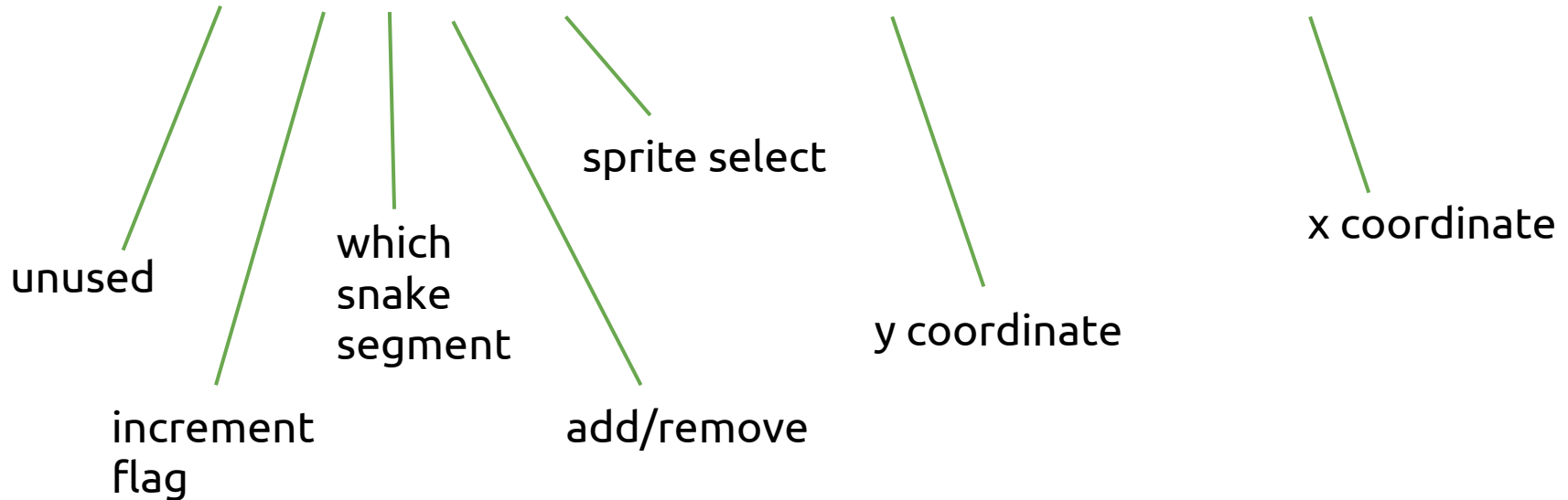
NES Controller Protocol



Design Architecture

Writing to hardware

00001011010110101010101010101010



Game Software

```
void updateSnake(struct Snake snake[], int xCoord, int yCoord)
{
    int player, struct SnakeInfo = info;
    //printf("Updating snake\n");
    // Remove tail first
    removeSnakePiece(player, snake[info-tail].xCoord,
    int tail_old_x = snake[info-tail-1].xCoord;
    int tail_old_y = snake[info-tail-1].yCoord;
    //Then update snake
    updateBodySnake, info;
    snake[0].xCoord = xCoord;
    snake[0].yCoord = yCoord;
    int tail_new_x = snake[info-tail-1].xCoord;
    int tail_new_y = snake[info-tail-1].yCoord;
    int tail_sprite = -1;
    if (tail_old_x != tail_new_x){
        if (tail_old_y == tail_new_y){
            //turned up
            tail_sprite = SNAKE_TAIL_UP;
        } else if (tail_old_y < tail_new_y){
            //formed down
            tail_sprite = SNAKE_TAIL_DOWN;
        } else {
            //same direction, which way?
        }
    }
}
```

list.h

writeToHW

```
void draw_winner(int winner_id){
    short i;
    short y = 10;
    for(i = 15; i < 25; i++){
        removeSnakePiece(PLAYER1, i, y);
        removeSnakePiece(PLAYER2, i, y);
        removeTilePiece(i, y);
    }
    if(winner_id == 1 && PLAYER1 == 1){
        draw_P1_wins();
    } else if(winner_id == 1 && PLAYER1 == 2){
        draw_P2_wins();
    } else if(winner_id == 2 && PLAYER2 == 2){
        draw_P2_wins();
    } else if(winner_id == 2 && PLAYER2 == 1){
        draw_P1_wins();
    } else if(winner_id == 3){
        draw_tie();
    }
}

void initPowUps(){
    initFood();
    initSpeed();
    initFreeze();
    initEdwards();
}
```

main.c

```
    }
    food_index++;
    food_count++;
    return 1;
}

void removeFood(int index){
    //printf("Removing food\n");
    food[index].enable = 0;
    removeTilePiece( food[index].xCoord,
    food_count--;
}

void shuffle_food(int n){
    int i;
    for(i = 0; i < n; i++){
        int index = PRNG(n);
        struct Food temp = food[index];
        food[index] = food[i];
        food[i] = temp;
    }
}
```

food.h

```
0; j < MAX_POWERUP_SIZE; j++){
    if(speed[j].enable){
        if(snake[0].xCoord == speed[j].xCoord &&
        snake[0].yCoord == speed[j].yCoord){
            //printf("Eating Speed!\n");
            removeSpeed(j);
            info->speed_enabled = 1;
            info->speed_count = 0;
            if(player == PLAYER1){
                PLAYER1_SLEEP_CYCLES = SPEED_SLEEP_CYCLE /
                SLEEP_TIME;
            } else{
                PLAYER2_SLEEP_CYCLES = SPEED_SLEEP_CYCLE /
                SLEEP_TIME;
            }
        }
    }
}
```

speed.h

```
!Freeze){
    //printf("Initialzing freeze\n");
    int i;
    int j;
    int count = 0;
    for(i = 0; i < X_LEN; i++){
        for(j = 0; j < Y_LEN; j++){
            if(board[i][j] == 1){
                freeze[count].enable = 0;
                freeze[count].xCoord = i;
                freeze[count].yCoord = j;
                count++;
            }
        }
    }
    shuffle_freeze(freeze_MAX_POWERUP_SIZE);
}
```

freeze.h

```
int remove_at = 0;
if(edwards_index == 0){
    remove_at = MAX_POWERUP_SIZE-1;
} else{
    remove_at = edwards_index - 1;
}

if(edwards[remove_at].type == 0){
    setTileSnake, info->tail;
} else if(edwards[remove_at].type == 1){
    if(PLAYER1 == 1 && PLAYER2 == 2){
        PLAYER1 = 1;
        PLAYER2 = 2;
    } else{
        PLAYER1 = 2;
        PLAYER2 = 1;
    }
} else{
    updateBorder();
}
info->has_edwards = 0;
```

edwards.h

brick.h

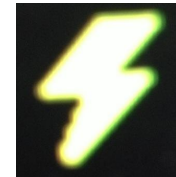
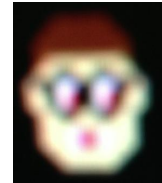
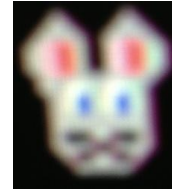
powboard.h

audio.h

Gameplay

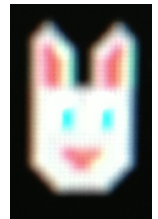
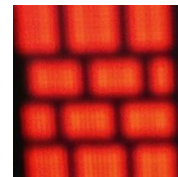
Edibles

- food (rabbits + mice)
- speed power-up
- freeze power-up
- edwards power-up

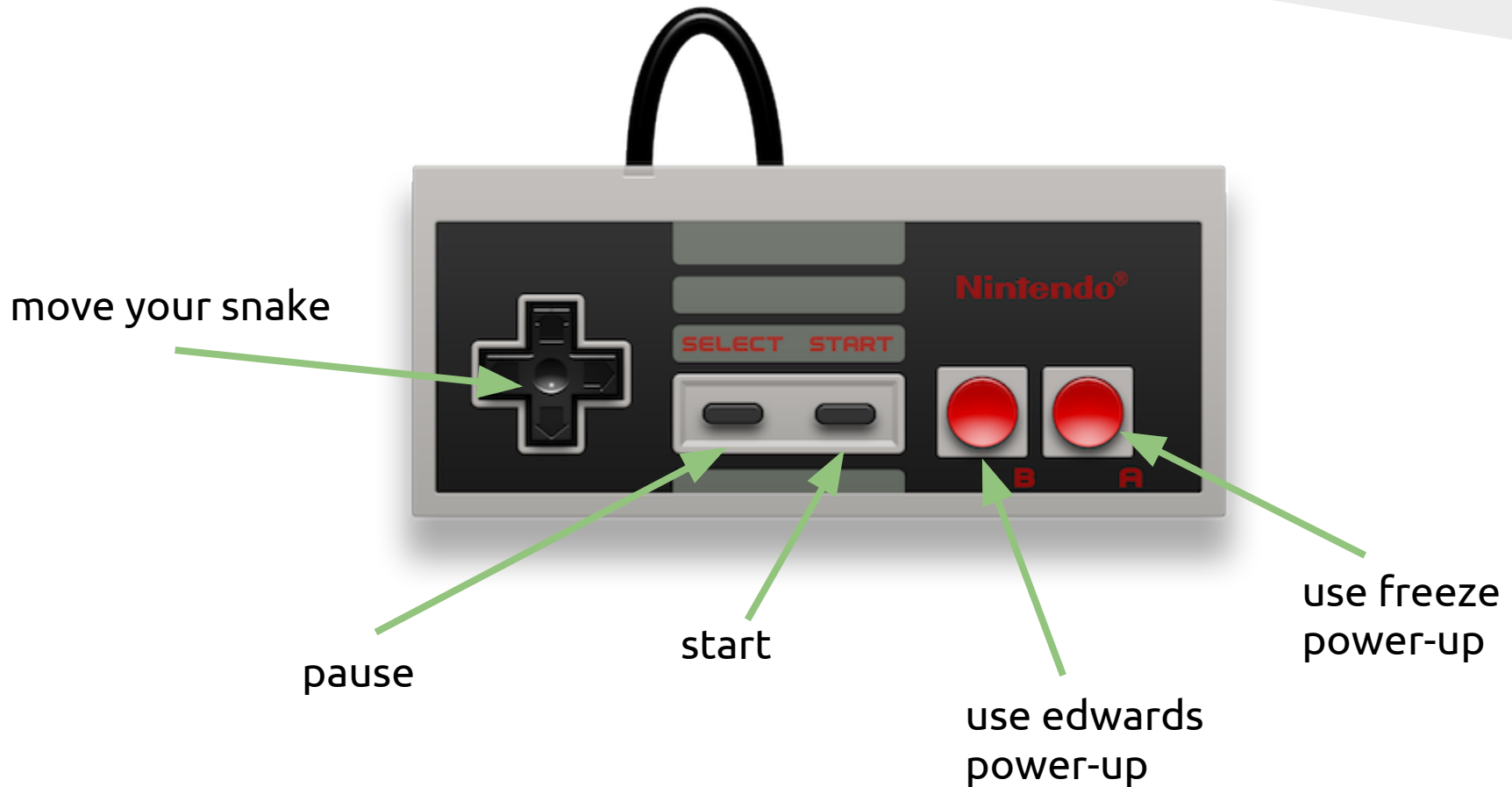


Obstacles

- bricks + walls
- your own tail
- other players



Controlling Your Snake



Challenges

- **Connecting NES controllers**
 - Using universal I/O pins
 - Have their own serial-bit protocol
- **Tracking snakes' locations**
 - Snakes are just linked lists
 - Software knows where everything is
 - Hardware just draws it
- **Efficient storage of sprites**
 - Binary encodings, broken up by color
- **VGA Raster**
 - Unexpected colorings

Summary

Lessons Learned:

- Working with embedded systems is tricky, yet rewarding
 - Lots of possibilities and methods of implementation
 - ...Which means a lot of learning from your own mistakes

Completed Game:

- Brought Snake into the 21st century
 - Added in idea of NES controllers
 - Two players (or more)!



Thanks for Listening!