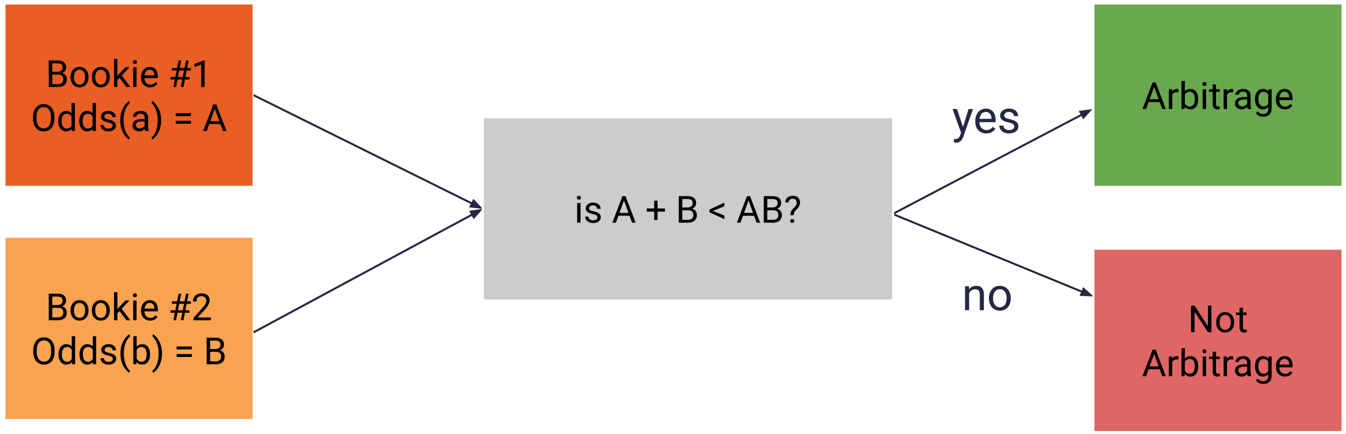
Sports Arbitrage Bettor

CSEE 4840: Embedded Systems

Team: Brennan McManus, Shivan Mukherjee, Jonathan Nalikka, Chelsea Soemitro, Shreya Somayajula

What is Arbitrage?

- Bettors place bets provided by bookmakers (bookies)
- Bettor can place multiple bets on *same* event from different bookies to guarantee profit, no matter the outcome of the event.
- Determined using simple comparison calculation:



Our Project

Detect combinations of bets on NBA games that result in guaranteed profit – i.e. are **arbitrage opportunities**.

Workflow

software

1

SCRAPE BOOKIES

Use Python API to scrape various bookie websites

2

PARSE RAW DATA

Transform data from API (assign IDs, convert to fixed-point)

hardware

3

ARBITRAGE DETECTION

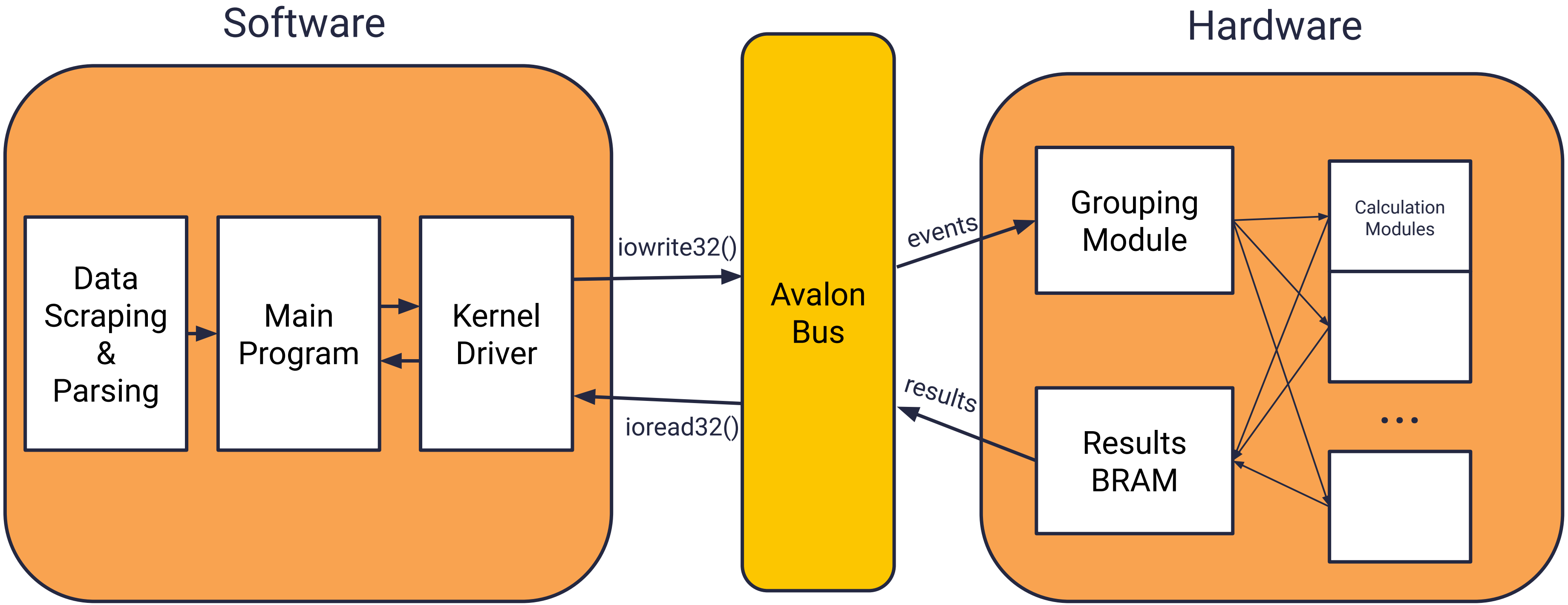
Send data to hardware to perform arbitrage detection

software

4

USER INTERFACE

Retrieve arbitrage results and calculate profit in software


Software
Hardware
Data Scraping & Parsing
Main Program
Kernel Driver
iowrite32()
ioread32()
Avalon Bus
events
results
Grouping Module
Results BRAM
Calculation Modules
...

Software Parsing: Data

the
odds-api


```json
{
    "id": "a00f81ba5c58d9646d7cfc95a3ce3904",
    "sport_key": "basketball_nba",
    "sport_title": "NBA",
    "commence_time": "2024-04-11T00:10:00Z",
    "home_team": "Milwaukee Bucks",
    "away_team": "Orlando Magic",
    "bookmakers": [
        {
            "key": "draftkings",
            "title": "DraftKings",
            "last_update": "2024-04-10T18:07:19Z",
            "markets": [
                {
                    "key": "h2h",
                    "last_update": "2024-04-10T18:07:19Z",
                    "outcomes": [
                        {
                            "name": "Milwaukee Bucks",
                            "price": 2.0
                        },
                        {
                            "name": "Orlando Magic",
                            "price": 1.83
                        }
                    ]
                }
```



```
Game ID,Home Team,Away Team,Bookmaker ID,Bookmaker Title,Outcome Name,Outcome Price
0,Cleveland Cavaliers,Boston Celtics,0,DraftKings,Boston Celtics,1.27
0,Cleveland Cavaliers,Boston Celtics,0,DraftKings,Cleveland Cavaliers,3.95
0,Cleveland Cavaliers,Boston Celtics,1,FanDuel,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,1,FanDuel,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,2,BetOnline.ag,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,2,BetOnline.ag,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,3,LowVig.ag,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,3,LowVig.ag,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,4,Caesars,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,4,Caesars,Cleveland Cavaliers,3.78
0,Cleveland Cavaliers,Boston Celtics,5,SuperBook,Boston Celtics,1.29
0,Cleveland Cavaliers,Boston Celtics,5,SuperBook,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,6,Bovada,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,6,Bovada,Cleveland Cavaliers,3.8
0,Cleveland Cavaliers,Boston Celtics,7,WynnBET,Boston Celtics,1.29
0,Cleveland Cavaliers,Boston Celtics,7,WynnBET,Cleveland Cavaliers,3.85
0,Cleveland Cavaliers,Boston Celtics,8,BetMGM,Boston Celtics,1.27
0,Cleveland Cavaliers,Boston Celtics,8,BetMGM,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,9,MyBookie.ag,Boston Celtics,1.29
0,Cleveland Cavaliers,Boston Celtics,9,MyBookie.ag,Cleveland Cavaliers,3.8
0,Cleveland Cavaliers,Boston Celtics,10,BetRivers,Boston Celtics,1.27
0,Cleveland Cavaliers,Boston Celtics,10,BetRivers,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,11,Unibet,Boston Celtics,1.27
0,Cleveland Cavaliers,Boston Celtics,11,Unibet,Cleveland Cavaliers,3.9
0,Cleveland Cavaliers,Boston Celtics,12,BetUS,Boston Celtics,1.28
0,Cleveland Cavaliers,Boston Celtics,12,BetUS,Cleveland Cavaliers,3.85
1,Dallas Mavericks,Oklahoma City Thunder,0,DraftKings,Dallas Mavericks,1.83
1,Dallas Mavericks,Oklahoma City Thunder,0,DraftKings,Oklahoma City Thunder,2.0
```

Software Parsing: Bookie Mapping

Bookie Name	Bookie ID
DraftKings	0
FanDuel	1
BetOnline.ag	2
…	…
SuperBook	12

the most we've seen is 13
—can be represented in 4 bits!

Software Parsing: Fixed-Point Conversion

floating point

x.xxx

fixed point

2^{10} | 2^{9} | … | 2^{2} | 2^{1} | 2^{0} | 2^{-1} | 2^{-2} | 2^{-3} | 2^{-4} | 2^{-5} | 2^{-6} | 2^{-7} | 2^{-8} | 2^{-9}

Software-Hardware Interface: Representation

Event struct (32-bit):

```
typedef struct {
    uint32_t odds:        20;
    uint32_t game_id:     4;
    uint32_t bookie_id:   4;
    uint32_t outcome:     1;
    uint32_t unused:      3;
} arb_event_t;
```

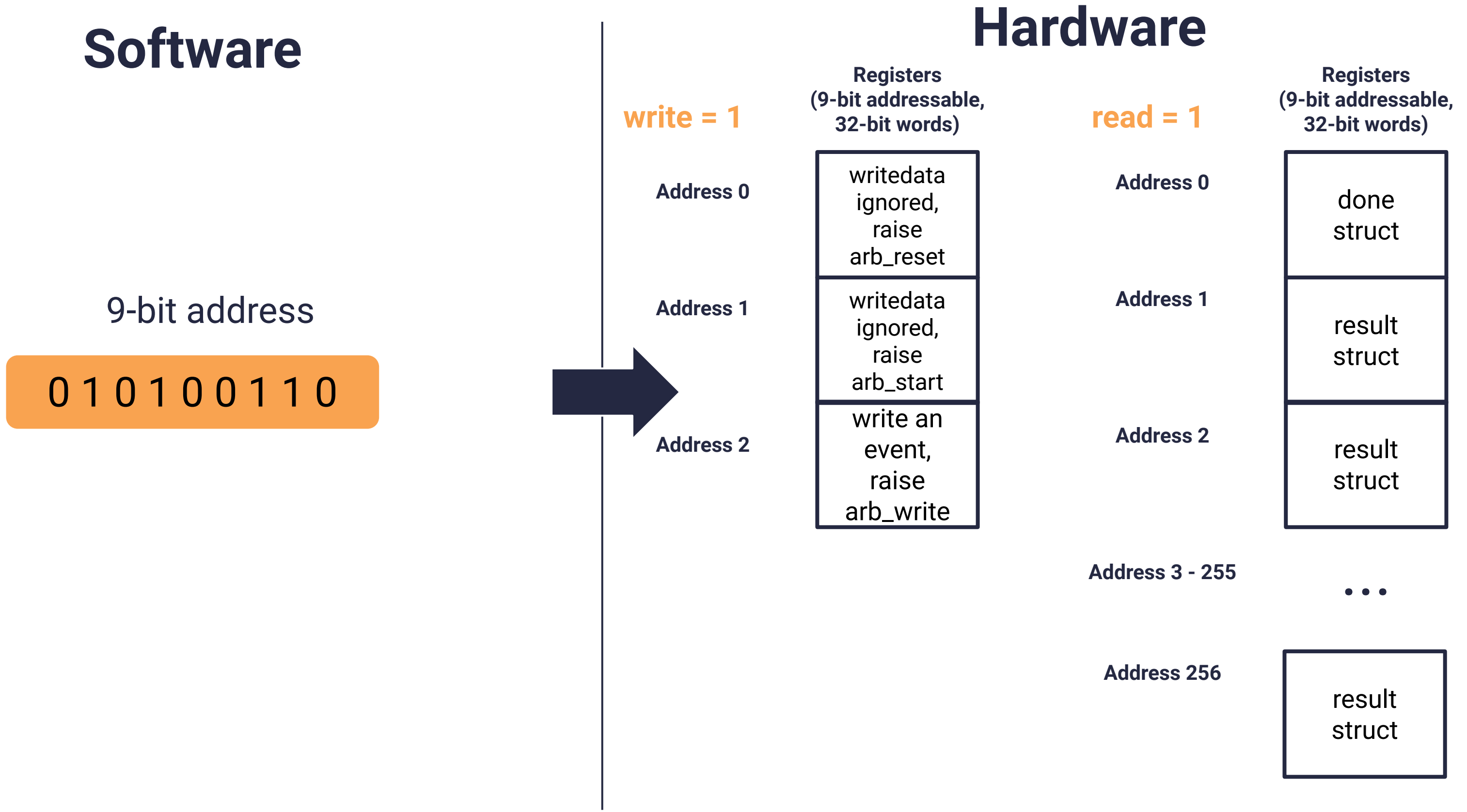
Result struct (32-bit):

```
typedef struct {
    uint32_t arb_prob:    20;
    uint32_t game_id:     4;
    uint32_t bookie_id_a: 4;
    uint32_t bookie_id_b: 4;
} arb_result_t;
```

Done struct (32-bit):

```
typedef struct {
    uint32_t done:            1;
    uint32_t result_count:    8;
    uint32_t padding:     23;
} arb_read_regs_t;
```

Software-Hardware Interface: Registers



Software-Hardware Interface: ioctls

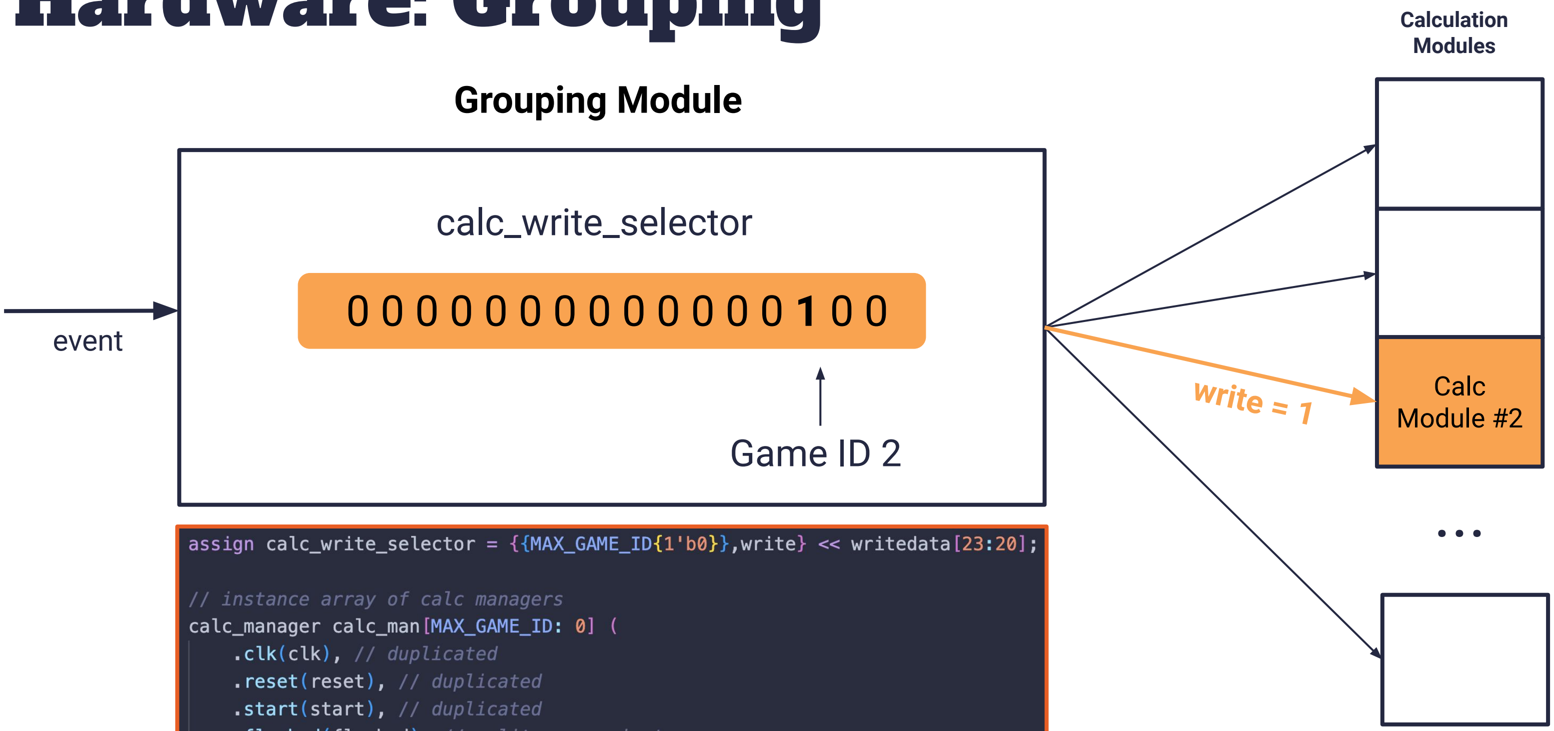
CALC_ARB_WRITE_EVENTS

```c
static void write_events(struct event_buf *buf)
{
    int i;

    // (1) send reset signal
    iowrite32(0, ((uint32_t *)dev.virtbase) + ARB_RESET_ADDR);

    // (2) write events
    for(i = 0; i < buf->len; i++) {
        iowrite32(*(uint32_t *)((buf->events_vec) + i),
                ((uint32_t *)dev.virtbase) + ARB_EVENT_WRITE_ADDR);
    }

    // (3) raise start
    iowrite32(0, ((uint32_t *)dev.virtbase) + ARB_START_ADDR);
}
```

CALC_ARB_READ_EVENTS

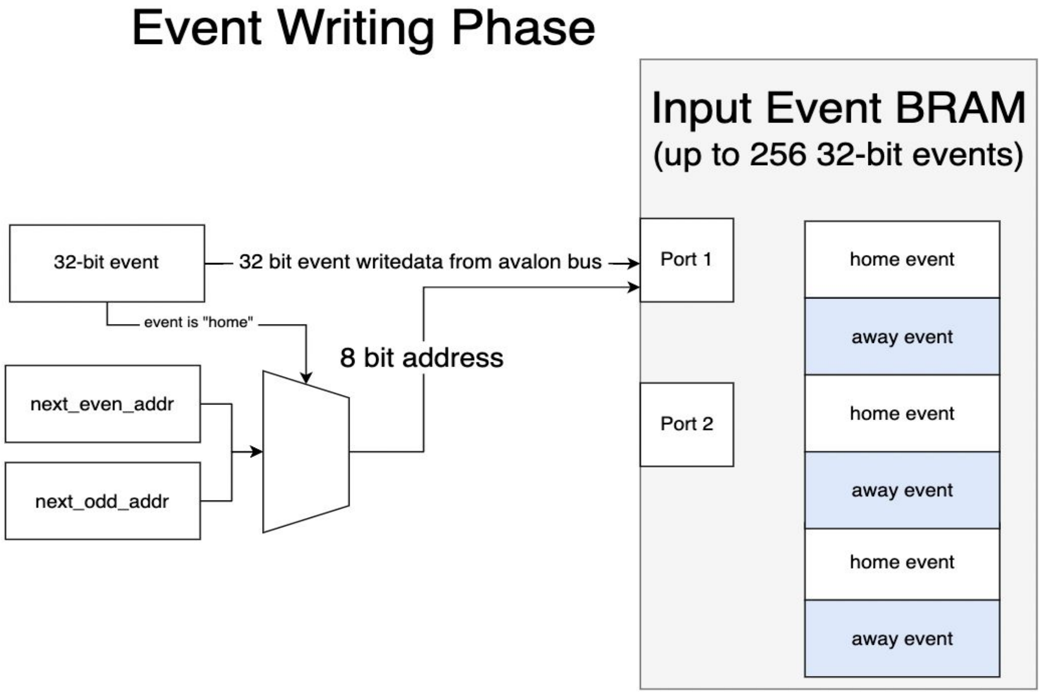
```c
static struct result_buf *read_result(void)
{
    arb_read_regs_t read_regs;
    int i;
    struct result_buf *results_buf;
    uint32_t readdata;

    // (1) poll for done
    while(1) {
        readdata = ioread32(((uint32_t *)dev.virtbase) + ARB_REGS_ADDR);
        read_regs = *((arb_read_regs_t *) &readdata);

        if (read_regs.done)
            break;
    }

    results_buf = kmalloc(sizeof(int) + read_regs.result_count * sizeof(arb_result_t), GFP_KERNEL);
    results_buf->len = read_regs.result_count;

    // (2) read results structs
    for (i=0; i < results_buf->len; i++) {
        uint32_t readdata = ioread32(((uint32_t *)dev.virtbase) + ARB_RESULT_READ_ADDR(i));
        results_buf->arbs_vec[i] = *((arb_result_t *) &readdata);
    }

    return results_buf;
}
```

Hardware: Grouping

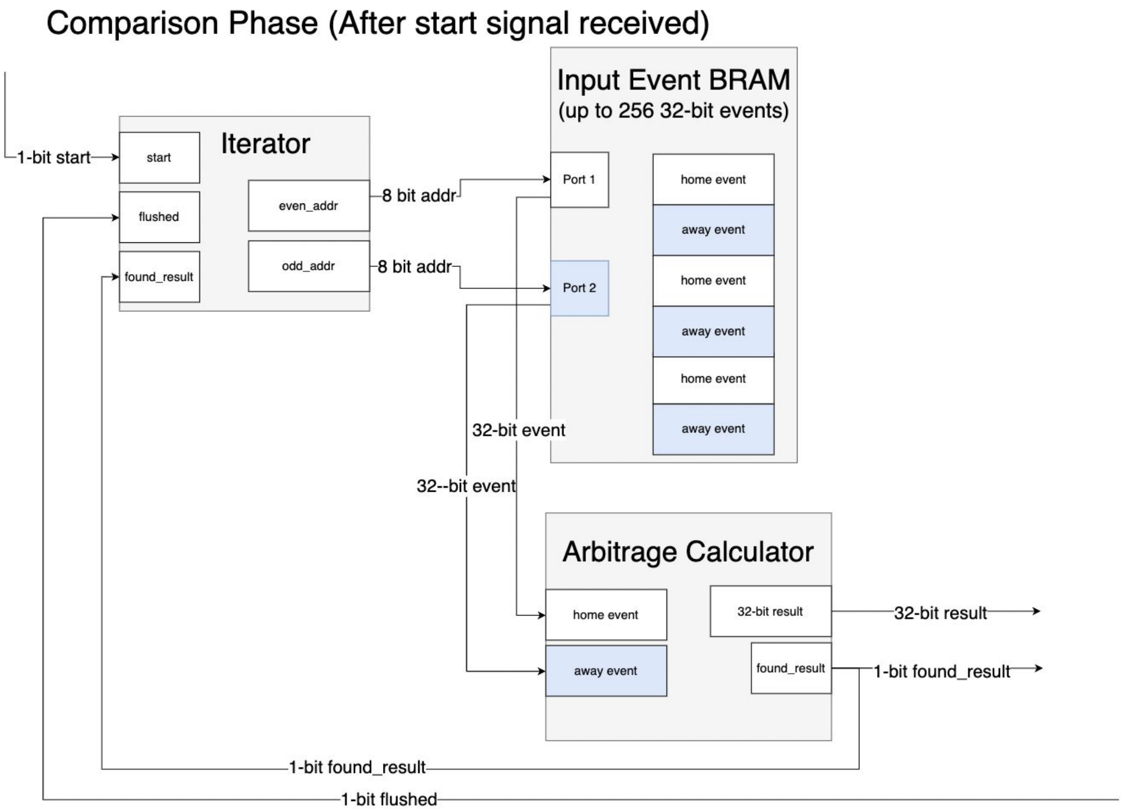



```
assign calc_write_selector = {{MAX_GAME_ID{1'b0}},write} << writedata[23:20];

// instance array of calc managers
calc_manager calc_man[MAX_GAME_ID: 0] (
    .clk(clk), // duplicated
    .reset(reset), // duplicated
    .start(start), // duplicated
    .flushed(flushed), // split across instances
    .write(calc_write_selector), // split across instances
    .writedata(writedata),  // duplicated
    .found_result(found_result), // split across instances
    .done(calc_done), // split across instances
    .result(result) // split across instances
);
```

Hardware Calculation: Event Writing Phase



Hardware Calculation: Comparison Phase

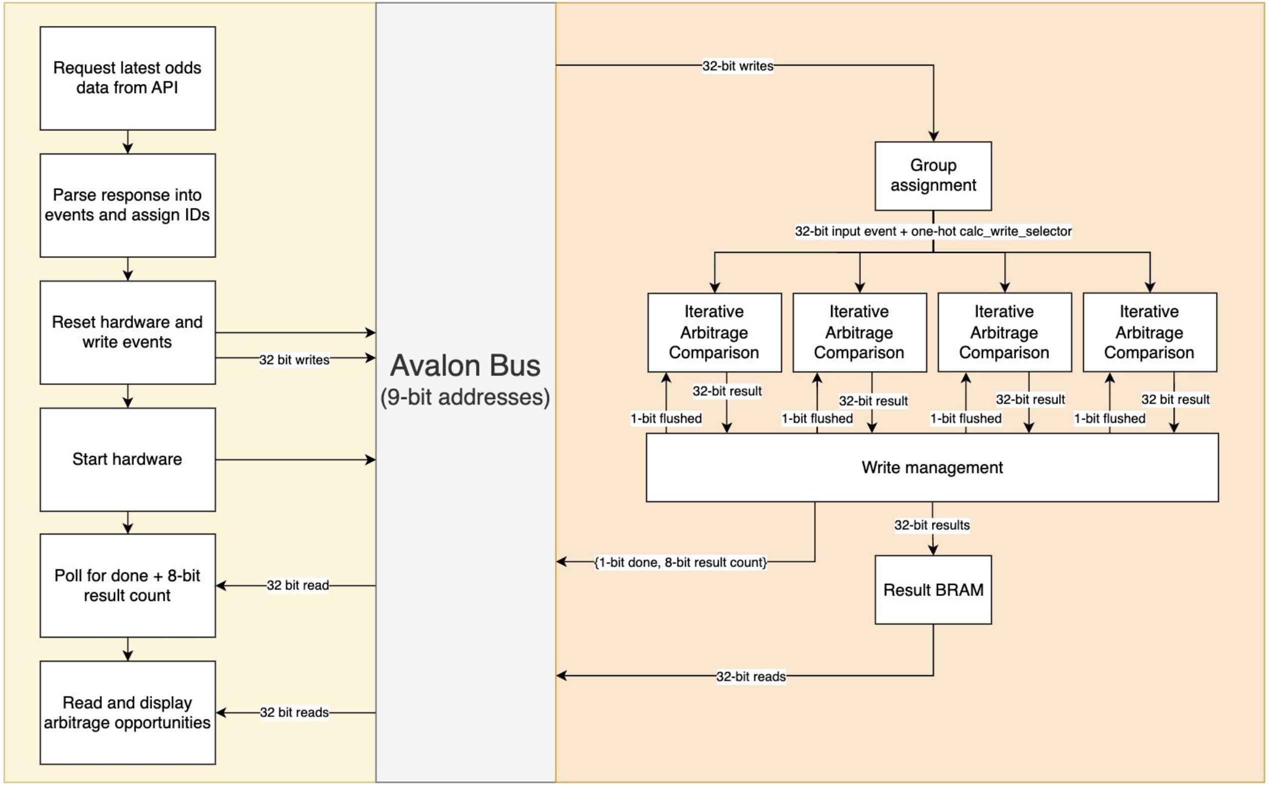


Hardware Calculation: calc_odds

Core Logic

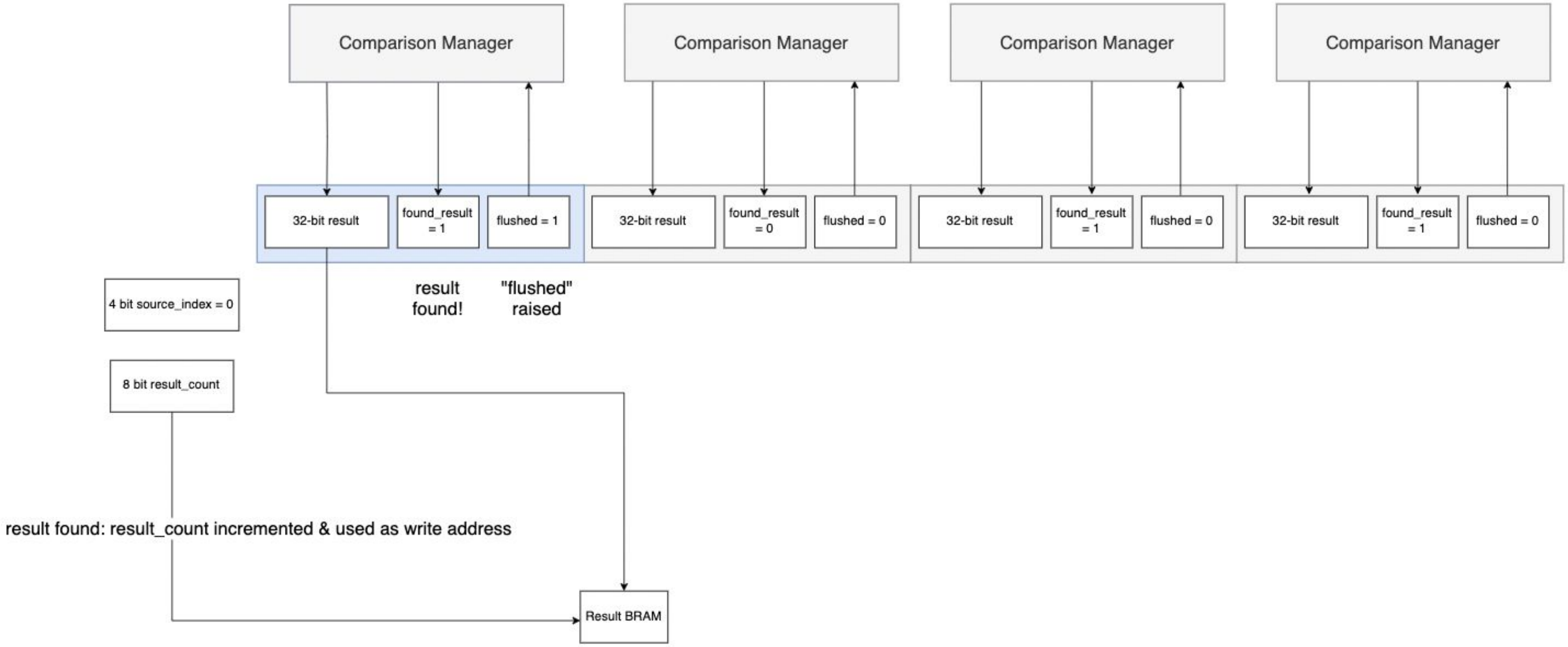
```
assign a20 = a[19:0];
assign b20 = b[19:0];

assign ab = a20 * b20;
assign aplusb = {10'b0, a20 + b20, 10'b0};

always_comb begin
    if (aplusb < ab) begin
        found_result = 1;
        arb_prob = a20 + b20;
    end else begin
        found_result = 0;
        arb_prob = 0;
    end
end
```

Synchronization: A more detailed view

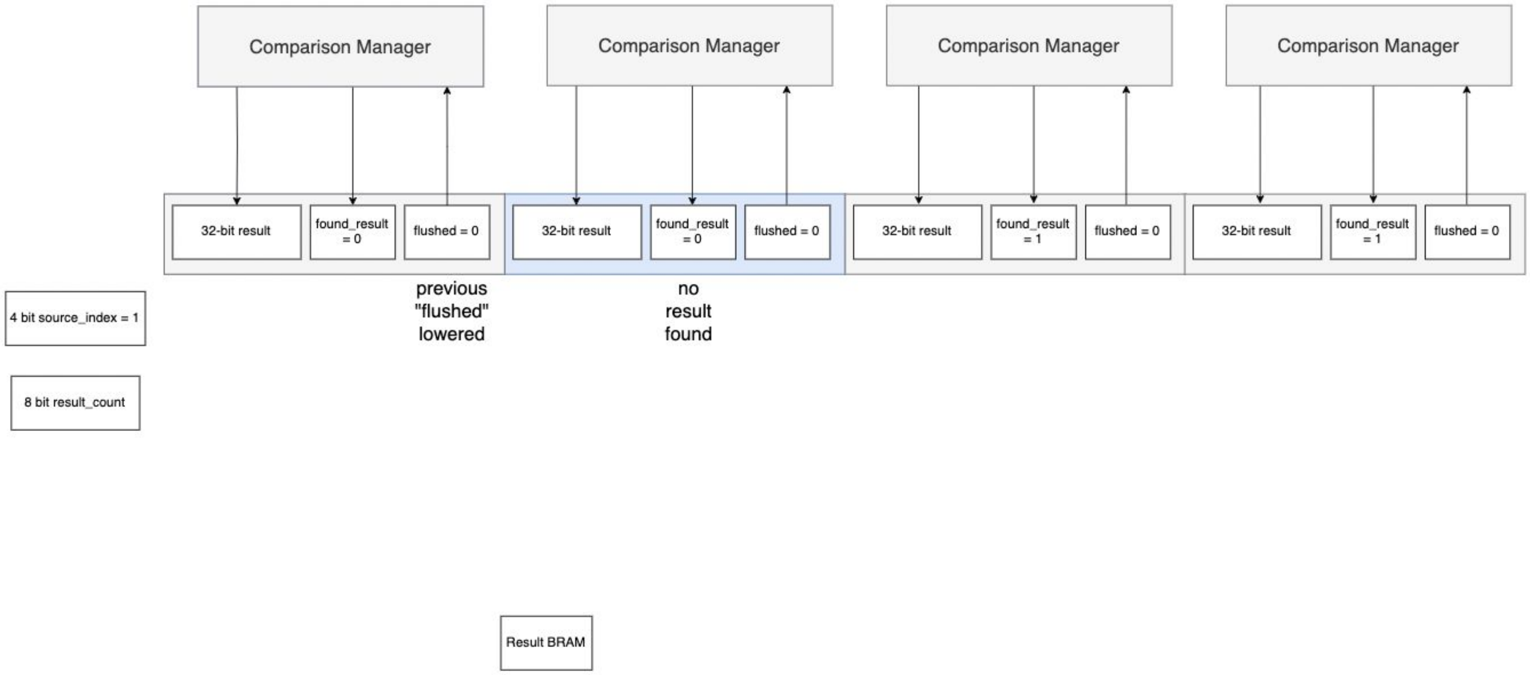


- Instance arrays help make things simpler!
- narrow vectors (clk, writedata) duplicated: each instance gets a copy
- wide vectors (calc_write_selector) distributed: each instance gets a slice

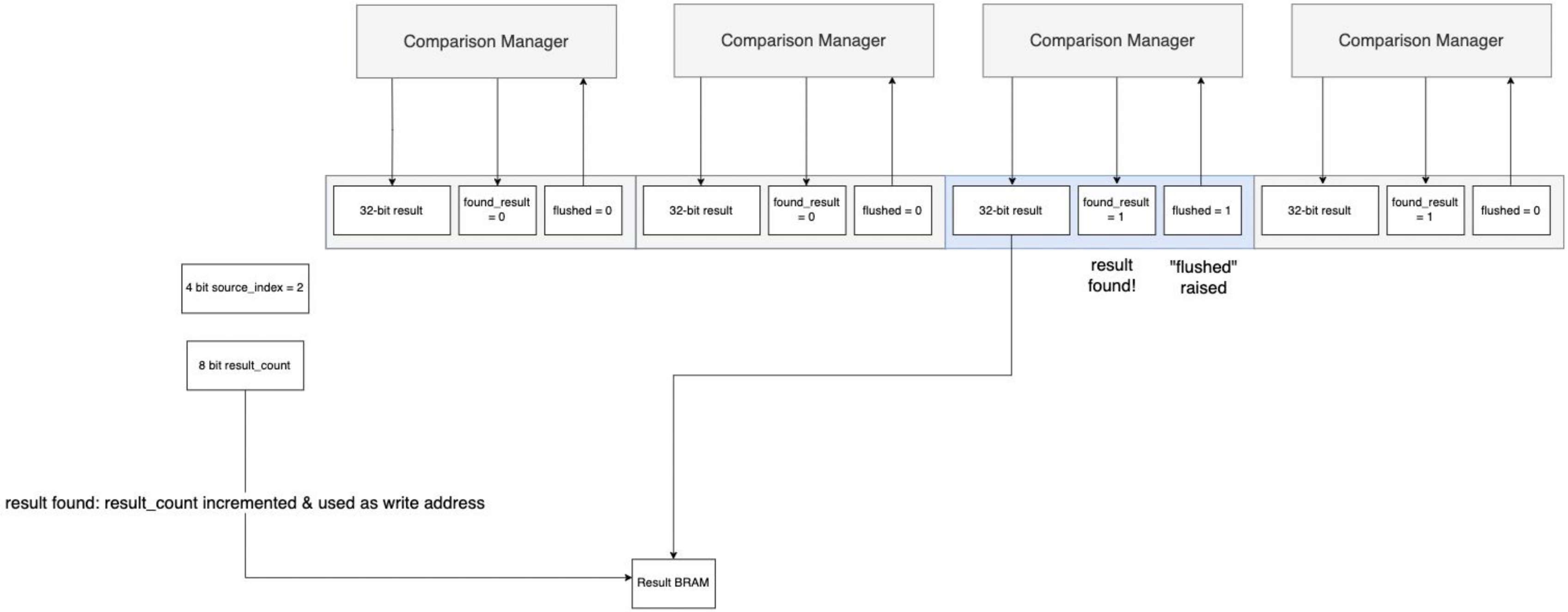
Synchronization: write_manager



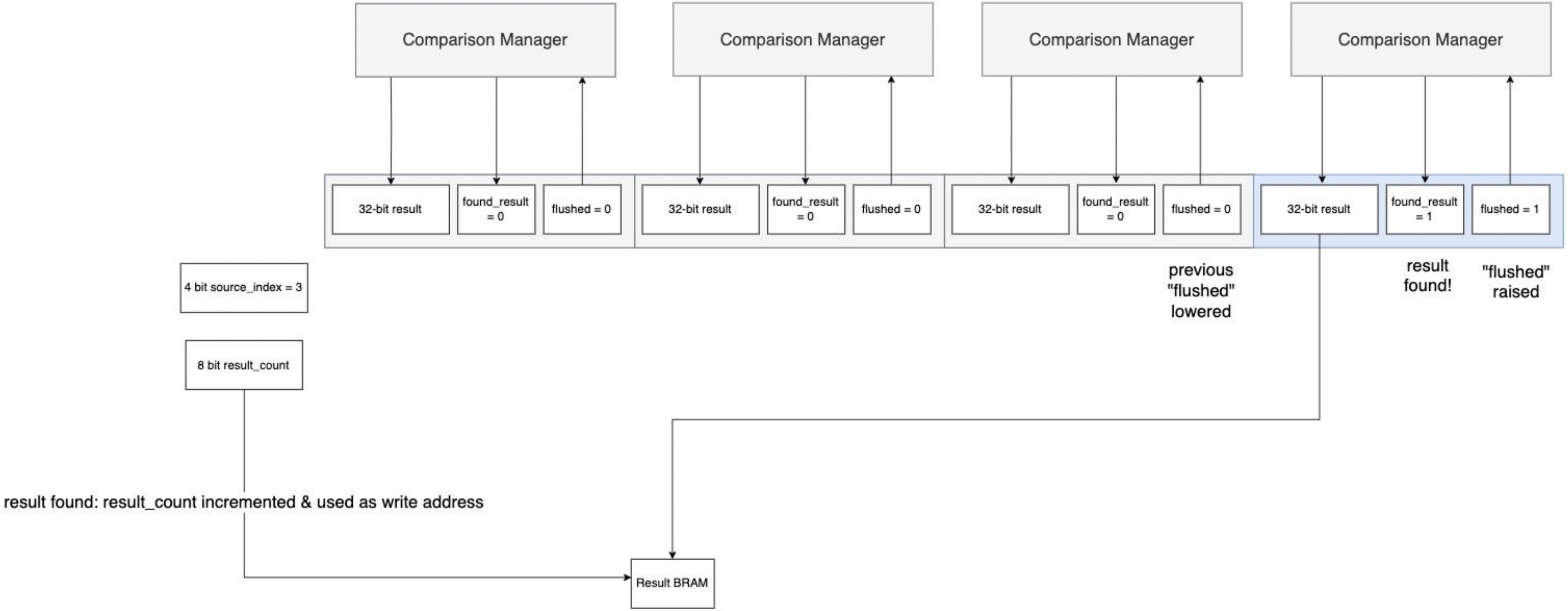
Synchronization: write_manager



Synchronization: write_manager



Synchronization: write_manager



Efficiency

Pure Python Implementation

Arbitrage Detection: 0.220 seconds

FPGA Implementation

Arbitrage Detection: 0.006 seconds

Demo Time!