Лабораторная работа №2	M3139	2022
Моделирование схем в Verilog	Некрасов Александр Павлович	

Цель работы: построение кэша и моделирование системы “процессор-кэш-память” на языке описания Verilog.

Инструментарий и требования к работе: Весь код пишется на языке Verilog, компиляция и симуляция – Icarus Verilog 10 и новее.

Описание

Моделирование работы системы “процессор-кэш-память”, определить процент кэш попаданий, и тактов

Вариант

1

Подсчет констант:

CACHE_LINE_SIZE = 16 байт

CACHE_LINE_COUNT = 64

CACHE_WAY = 2

CACHE_SETS_COUNT = 32

CACHE_TAG_SIZE = 10 бит

CACHE_SET_SIZE = log2(CACHE_LINE_COUNT) / CACHE_WAY = 5 бит

MEM_SIZE = 512 Кбайт = 512 * 1024 байт

CACHE_ADDR_SIZE = log2(MEM_SIZE) = 19 бит

CACHE_OFFSET_SIZE = CACHE_ADDR_SIZE - CACHE_SET_SIZE - CACHE_TAG_SIZE = 4 бита

ADDR1_BUS_SIZE = 15 бит

ADDR2_BUS_SIZE = 15 бит

DATA1_BUS_SIZE = 16 бит

DATA2_BUS_SIZE = 16 бит

CTR1_BUS_SIZE = 3 бита

CTR2_BUS_SIZE = 2 бита

Аналитическое решение:

```
#include <bits/stdc++.h>

using namespace std;
#define M 64
#define N 60
#define K 32

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int tkts = 0;
    int cache_hit = 0;
    int quers = 0;
    vector<vector<int>> CACHE1(32, vector<int>(2,
-1)); // хранит tag и dirty, размер - количество кеш
линий / 2
    vector<vector<int>> CACHE2(32, vector<int>(2,
-1)); // хранит tag и dirty, размер - количество кеш
линий / 2
    vector<int> old1(32); //  размер - количество кеш
линий / 2
    vector<int> old2(32); //  размер - количество кеш
линий / 2
    int f = 0;
    tkts += 1;
    int s = M * K + N * K * 2;
    tkts += 1;
    int time = 1;
    for (int i = 0; i < M; i++) {
        tkts += 2;
        for (int j = 0; j < N; j++) {
```

```
            tkts += 3;
            int t = M * K;
            for (int k = 0; k < K; k++) {
                tkts += 2;
                int set = ((f + k) >> 4) % 32; //
set вычисляем так, берем индекс в памяти, сдвигаем
на offset, и берем только принажлежащие сету
                tkts += 1;
                int tag = (f + k) >> 9; // уберем из
индекса set и offset
                tkts += 1;
                if (CACHE1[set][0] == tag) { //
попадание
                    tkts += 6; // отклик кеша при
попадании
                    cache_hit++;
                    quers++;
                    old1[set] = time++;
                } else if (CACHE2[set][0] == tag) {
// попадание
                    tkts += 6; // отклик кеша при
попадании
                    cache_hit++;
                    quers++;
                    old2[set] = time++;
                } else { // промах
                    tkts += 4; // отклик кеша при
промахе
                    tkts += 100; // отклик памяти
                    quers++;
                    if (old1[set] > old2[set]) {
                        if (CACHE2[set][1] == 1) {
// если линия была изменена, нужно записать
изменения в память, не влияет на попадания
                            tkts += 100; // отклик
памяти
                            CACHE2[set][1] = 0;
                        }
                        CACHE2[set][0] = tag;
                        old2[set] = time++;
                    } else {
                        if (CACHE1[set][1] == 1) {
                            tkts += 100; // отклик
```

```
памяти
                                CACHE1[set][1] = 0;
                            }
                            CACHE1[set][0] = tag;
                            old1[set] = time++;
                        }
                    }
                    set = ((t + j * 2) >> 4) % 32;
                    tag = (t + j * 2) >> 9;
                    if (CACHE1[set][0] == tag) {
                        tkts += 6; // отклик кеша при
попадании
                        cache_hit++;
                        quers++;
                        old1[set] = time++;
                    } else if (CACHE2[set][0] == tag) {
                        tkts += 6; // отклик кеша при
попадании
                        cache_hit++;
                        quers++;
                        old2[set] = time++;
                    }else{
                        quers++;
                        tkts += 4; // отклик кеша при
промахе
                        tkts += 100; // отклик памяти
                        if (old1[set] > old2[set]) {
                            if (CACHE2[set][1] == 1) {
                                tkts += 100; // отклик
памяти
                                CACHE2[set][1] = 0;
                            }
                            CACHE2[set][0] = tag;
                            old2[set] = time++;
                        } else {
                            if (CACHE1[set][1] == 1) {
                                tkts += 100; // отклик
памяти
                                CACHE1[set][1] = 0;
                            }
                            CACHE1[set][0] = tag;
                            old1[set] = time++;
                        }
```

```
                }
                t += N * 2;
                tkts += 1;
            }
            int set = ((s + j * 4) >> 4) % 32;
            int tag = (s + j * 4) >> 9;
            if (CACHE1[set][0] == tag) {
                tkts += 6; // отклик кеша при
попадании
                cache_hit++;
                quers++;
                CACHE1[set][1] = 1;
                old1[set] = time++;
            } else if (CACHE2[set][0] == tag) {
                tkts += 6; // отклик кеша при
попадании
                cache_hit++;
                quers++;
                CACHE2[set][1] = 1;
                old2[set] = time++;
            } else {
                tkts += 4; // отклик кеша при
промахе
                tkts += 100; // отклик памяти
                quers++;
                if (old1[set] > old2[set]) {
                    if (CACHE2[set][1] == 1) {
                        tkts += 100; // отклик
памяти
                        CACHE2[set][1] = 0;
                    }
                    CACHE2[set][0] = tag;
                    CACHE2[set][1] = 1;
                    old2[set] = time++;
                } else {
                    if (CACHE1[set][1] == 1) {
                        tkts += 100; // отклик
памяти
                        CACHE1[set][1] = 0;
                    }
                    CACHE1[set][0] = tag;
                    CACHE1[set][1] = 1;
                    old1[set] = time++;
```

```
                }
            }
        }
        f += K;
        s += N * 4;
        tkts += 2;
    }
    cout << tkts << endl;
    cout << cache_hit << " " << quers << " " <<
(double )cache_hit / quers << endl;
    int cache_hit_ans = 224175;
    cout << cache_hit_ans << " " << cache_hit_ans <<
" " << (double )224175 / quers << endl; // ответ
полученный при помощи модулирования работы кеша
    return 0;
}
```

Процент попаданий подсчитанный такой программой будет абсолютно точным, количество тактов можно посчитать только примерно, так как оно сильно зависит от реализации.

Получившиеся попадания = 228080 из 249600 запросов, т.е. 91.38%

Кол-во тактов = 4406538

Моделирование системы на Verilog

У нас есть три модуля, mem, cache, и CPU, mem поддерживает три запроса, C2_NOP (no operation), C2_READ_LINE передать в кеш линию с заданным адресом, C2_WRITE_LINE записать в память кеш линию в заданное место, на языке Verilog я написал обработку команд приходящих в mem в блоке always (за переключение тактов отвечает clk, @(posedge clk) ждет, пока clk переключится из 0 в 1):

```
always @(posedge clk) begin
        tD = D2;
```

```
        tC = C2;
        if (C2 == 2) begin
            for (i = 0; i < 100; i += 1) begin
                @(posedge clk);
            end
            addr[18:4] = A2;
            for (step = 0; step < 16; step += 2) begin
                if (step != 0) begin
                    @(posedge clk);
                end
                tD[7:0] = a[addr + step];
                tD[15:8] = a[addr + step + 1];
                tC = 1;
            end
            tC = 0;
        end else if (C2 == 3) begin
            for (i = 0; i < 100; i += 1) begin
                @(posedge clk);
            end
            addr[18:4] = A2;
            for (step = 0; step < 16; step += 2) begin
                @(posedge clk);
                tC = 1;
                a[addr + step] = D2[7:0];
                a[addr + 1 + step] = D2[15:8];
                while (write != 1) @(posedge clk);
            end
            tC = 0;
        end
    end
```

т.к. нужно промоделировать ожидание памяти в 100 тактов, сделаем это в for, когда по шине C2 приходит осмысленная команда, когда команда C2 = 2 - т.е. READ_LINE передаем по шине D2 за 8 тактов(т.к. длина кеш линии 16 байт а размер шины 16 бит) начиная с addr(адрес, полученный по A2), когда на шине C2 команда 3 - это WRITE_LINE, получаем по шине D2 за 8 тактов кеш линию с адресом addr. Cache должен обрабатывать однотипный

команды C1_READ8 - прочитать 8 бит, C1_READ16 и C1_READ32 прочитать 16 и 32 бита соответственно, C1_WRITE8, C1_WRITE16, C1_WRITE32 - записать 8, 16, и 32 бита соответственно, чтоб прочитать или записать что то в кеш, нужно сначала получить требуемую кеш линию, либо она уже есть в кеше, т.е. произошло кеш попадание, либо нужно прочитать линию из памяти. Когда хотим прочитать из памяти, заменять нужно либо invalid линию с соответствующим set, либо более старую линию, при этом, если линия dirty(отличается от данных в памяти) ее нужно предварительно записать в память, это действие общее для всех команд, так как в данной программе все запросы попадают в одну кэш линию, поэтому можно промоделировать всю работу с памятью, когда приходит осмысленная команда(просто записать линию в память, если она dirty и прочитать новую), когда запрос на один или два байта передаем все по шине D1 за один такт, если же запрос READ32 или WRITE32 нужно будет передавать за два такта, в модуле CPU напишем саму программу, посчитаем количество тактов, и сделаем запросы в cache.

Промоделируем систему на Verilog, и получим количество тактов, и кэш попаданий:
cache_hits: 224175 calcer: 6227032, где cache_hits - кэш попадания, calcer - количество тактов

Количество тактов отличается от аналитического решения в полтора раза, процент кэш попаданий равен 89.81% что отличается от верного количества на 1.56%

Листинг кода:

```
`define M 64
```

```
`define N 60
`define K 32


module mem (input[14:0] A2, inout[15:0] D2, inout[1:0] C2, input clk,
m_dump, reset, write); // ADDR2_BUS_SIZE = 15 DATA2_BUS_SIZE = 16
CTR2_BUS_SIZE = 2
    reg[15:0] tD;
    reg[1:0] tC;
    integer SEED = 225526;
    reg[7:0] a[0:512 * 1024]; // MEM_SIZE = 512 килобайт
    integer i = 0;
    integer step = 0;
    integer _wait = 0;
    reg[18:0] addr = 0; // CACHE_ADDR_SIZE = 19

    initial begin
        for (i = 0; i < 512 * 1024; i += 1) begin
            a[i] = $random(SEED)>>16;
        end

        // for (i = 0; i < 512 * 1024; i += 1) begin
        // $display("[%d] %d", i, a[i]);
        // end

    end

    always @(posedge reset) begin
        for (i = 0; i < 512 * 1024; i += 1) begin
            a[i] = $random(SEED)>>16;
        end
    end

    always @(posedge m_dump) begin
        for (i = 0; i < 512 * 1024; i += 1) begin
            $display("[%d] %d", i, a[i]);
        end
    end

    always @(posedge clk) begin
        tD = D2;
```

```
        tC = C2;
        if (C2 == 2) begin
            $display("C2: %d A2: %d", C2, A2);
            for (i = 0; i < 100; i += 1) begin
                @(posedge clk);
            end
            addr[18:4] = A2;
            for (step = 0; step < 16; step += 2) begin // CACHE_LINE_SIZE =
16 байт
                if (step != 0) begin
                    @(posedge clk);
                end
                tD[7:0] = a[addr + step];
                tD[15:8] = a[addr + step + 1];
                tC = 1;
            end
            tC = 0;
        end else if (C2 == 3) begin
            $display("C2: %d A2: %d", C2, A2);
            for (i = 0; i < 100; i += 1) begin
                @(posedge clk);
            end
            addr[18:4] = A2;
            for (step = 0; step < 16; step += 2) begin
                @(posedge clk);
                tC = 1;
                a[addr + step] = D2[7:0];
                a[addr + 1 + step] = D2[15:8];
                while (write != 1) @(posedge clk);
            end
            tC = 0;
        end
    end

    assign D2 = (write == 0)? tD: 16'bzzzzzzzzzzzzzzzz;
    assign C2 = (write == 0)? tC: 2'bzz;
endmodule


module cache(output[31:0] cache_hits, input[14:0] A1, output[14:0] A2,
inout[15:0] D1, D2, inout[2:0] C1, inout[1:0] C2, input clk, c_dump, reset,
```

```
output write, read);
    reg write = 1;
    reg read = 1;
    reg[31:0] cache_hits = 0;
    reg[14:0] tA1;
    reg[15:0] tD1;
    reg[2:0] tC1;
    reg[2:0] mC1;
    reg[14:0] tA2;
    reg[15:0] tD2;
    reg[1:0] tC2;
    reg[11:0] cache_way_tag1[0:31];
    reg[11:0] cache_way_tag2[0:31];
    reg[7:0] cache_way1[0:31][0:15];
    reg[7:0] cache_way2[0:31][0:15];
    reg[31:0] cache_old1[0:31];
    reg[31:0] cache_old2[0:31];
    reg[31:0] _time = 1;
    integer i = 0;
    integer j = 0;
    integer cache_hit = 0;
    reg[9:0] tag;
    reg[4:0] set;
    reg[3:0] offset;
    reg[14:0] addr;

    initial begin
        for (i = 0; i < 32; i += 1) begin
            cache_way_tag1[i][11] = 0;
            cache_way_tag2[i][11] = 0;
            cache_way_tag1[i][10] = 0;
            cache_way_tag2[i][10] = 0;
            cache_old1[i] = 0;
            cache_old2[i] = 0;
        end

    end

    always @(posedge reset) begin
        for (i = 0; i < 32; i += 1) begin
            cache_way_tag1[i][11] = 0;
```

```
            cache_way_tag2[i][11] = 0;
            cache_way_tag1[i][10] = 0;
            cache_way_tag2[i][10] = 0;
            cache_old1[i] = 0;
            cache_old2[i] = 0;
        end
    end

    always @(posedge c_dump) begin
        $display("Way 1:");
        for (i = 0; i < 32; i += 1) begin
            $write("Line [%d]:", i);
            for (j = 11; j >= 0; j -= 1) begin
                $write("%d", cache_way_tag1[i][j]);
            end
            for (j = 0; j < 16; j += 1) begin
                $write(" %d ", cache_way1[i][j]);
            end
            $display();
        end
        $display("Way 2:");
        for (i = 0; i < 32; i += 1) begin
            $write("Line [%d]:", i);
            for (j = 11; j >= 0; j -= 1) begin
                $write("%d", cache_way_tag2[i][j]);
            end
            for (j = 0; j < 16; j += 1) begin
                $write(" %d ", cache_way2[i][j]);
            end
            $display();
        end
    end

    always @(posedge clk) begin
        cache_hit = 0;
        tD1 = D1;
        tC1 = C1;
        tA2 = A2;
        tD2 = D2;
        tC2 = C2;
        if (C1 != 0) begin
```

```
            // $display("C1: %d tag: %d set: %d offset: %d A1: %d", mC1,
tag, set, offset, A1);
            mC1 = C1;
            tag = A1[14:5];
            set = A1[4:0];
            addr = A1;
            for (i = 0; i < 4; i += 1) begin
                @(posedge clk);
                offset = A1[3:0];
            end
            if (mC1 == 7) begin
                $display("C1: %d tag: %d set: %d offset: %d", mC1, tag, set,
offset);
            end
            tC1 = 0;
            read = 0;
            @(posedge clk);
            read = 1;
            @(posedge clk);
            if (cache_way_tag1[set][11] == 1 && cache_way_tag1[set][9:0] ==
tag) begin
                cache_hit = 1;
                cache_hits += 1;
                for (i = 0; i < 2; i += 1) begin
                    @(posedge clk);
                end
            end
            if (cache_way_tag2[set][11] == 1 && cache_way_tag2[set][9:0] ==
tag) begin
                cache_hit = 2;
                cache_hits += 1;
                for (i = 0; i < 2; i += 1) begin
                    @(posedge clk);
                end
            end
            if (cache_hit == 0 && mC1 != 4) begin
                if (cache_old1[set] > cache_old2[set]) begin
                    cache_hit = 2;
                    if (cache_way_tag2[set][10] == 1) begin
                        tD2[7:0] = cache_way2[set][0];
                        tD2[15:8] = cache_way2[set][1];
```

```
                    tC2 = 3;
                    tA2[14:5] = cache_way_tag2[set];
                    tA2[4:0] = set;
                    @(posedge clk);
                    write = 0;
                    while (C2 != 1) @(posedge clk);
                    tC2 = 1;
                    for (i = 2; i < 16; i += 2) begin
                        while(C2 != 1) @(posedge clk);
                        write = 1;
                        @(posedge clk);
                        tD2[7:0] = cache_way2[set][i];
                        tD2[15:8] = cache_way2[set][i + 1];
                    end
                    cache_way_tag2[set][10] = 0;
                    @(posedge clk);
                end
                tC2 = 2;
                tA2 = addr;
                @(posedge clk)
                write = 0;
                while (C2 != 1) @(posedge clk);
                tC2 = 1;
                cache_way2[set][0] = D2[7:0];
                cache_way2[set][1] = D2[15:8];
                for (i = 2; i < 16; i += 2) begin
                    @(posedge clk);
                    cache_way2[set][i] = D2[7:0];
                    cache_way2[set][i + 1] = D2[15:8];
                end
                write = 1;
                tC2 = 0;
                cache_way_tag2[set][9:0] = tag;
                cache_way2[set][10] = 0;
                cache_way_tag2[set][11] = 1;
            end else begin
                cache_hit = 1;
                if (cache_way_tag1[set][10] == 1) begin
                    tD2[7:0] = cache_way1[set][0];
                    tD2[15:8] = cache_way1[set][1];
                    tC2 = 3;
```

```
                    tA2[14:5] = cache_way_tag1[set];
                    tA2[4:0] = set;
                    @(posedge clk);
                    write = 0;
                    tC2 = 1;
                    for (i = 2; i < 16; i += 2) begin
                        while(C2 != 1) @(posedge clk);
                        write = 1;
                        @(posedge clk);
                        tD2[7:0] = cache_way1[set][i];
                        tD2[15:8] = cache_way1[set][i + 1];
                    end
                    cache_way_tag1[set][10] = 0;
                    @(posedge clk);
                end
                tC2 = 2;
                tA2 = addr;
                @(posedge clk);
                write = 0;
                while (C2 != 1) @(posedge clk);
                tC2 = 1;
                cache_way1[set][0] = D2[7:0];
                cache_way1[set][1] = D2[15:8];
                for (i = 2; i < 16; i += 2) begin
                    @(posedge clk);
                    cache_way1[set][i] = D2[7:0];
                    cache_way1[set][i + 1] = D2[15:8];
                end
                write = 1;

                @(posedge clk);
                tC2 = 0;
                cache_way_tag1[set][9:0] = tag;
                cache_way_tag1[set][11] = 1;
            end
        end
        if (cache_hit == 1) begin
            cache_old1[set] = _time;
            _time += 1;
            if (mC1 == 7) begin
                cache_way_tag1[set][10] = 1;
```

```
                cache_way1[set][offset] = D1[7:0];
                cache_way1[set][offset + 1] = D1[15:8];
                read = 0;
                tC1 = 7;
                @(posedge clk);
                read = 1;
                cache_way1[set][offset + 2] = D1[7:0];
                cache_way1[set][offset + 3] = D1[15:8];
            end
            if (mC1 == 1) begin
                read = 0;
                tD1[7:0] = cache_way1[set][offset];
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 2) begin
                read = 0;
                tD1[7:0] = cache_way1[set][offset];
                tD1[15:8] = cache_way1[set][offset + 1];
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 3) begin
                read = 0;
                tD1[7:0] = cache_way1[set][offset];
                tD1[15:8] = cache_way1[set][offset + 1];
                tC1 = 7;
                @(posedge clk);
                tD1[7:0] = cache_way1[set][offset + 2];
                tD1[15:8] = cache_way1[set][offset + 3];
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 4) begin
                read = 0;
                @(posedge clk);
                cache_way_tag1[set][11] = 0;
                cache_old1[set] = 0;
                tC1 = 0;
```

```
                    @(posedge clk);
                    read = 1;
                end
                if (mC1 == 5) begin
                    cache_way1[set][offset] = D1[7:0];
                    read = 0;
                    tC1 = 0;
                    @(posedge clk);
                    read = 1;
                end
                if (mC1 == 6) begin
                    cache_way1[set][offset] = D1[7:0];
                    cache_way1[set][offset + 1] = D1[15:8];
                    read = 0;
                    tC1 = 0;
                    @(posedge clk);
                    read = 1;
                end
            end else begin
                cache_old2[set] = _time;
                _time += 1;
                if (mC1 == 7) begin
                    cache_way_tag2[set][10] = 1;
                    cache_way2[set][offset] = D1[7:0];
                    cache_way2[set][offset + 1] = D1[15:8];
                    read = 0;
                    tC1 = 7;
                    @(posedge clk);
                    read = 1;
                    cache_way2[set][offset + 2] = D1[7:0];
                    cache_way2[set][offset + 3] = D1[15:8];
                end
                if (mC1 == 1) begin
                    read = 0;
                    tD1[7:0] = cache_way2[set][offset];
                    tC1 = 7;
                    @(posedge clk);
                    read = 1;
                end
                if (mC1 == 2) begin
                    read = 0;
```

```
                tD1[7:0] = cache_way2[set][offset];
                tD1[15:8] = cache_way2[set][offset + 1];
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 3) begin
                read = 0;
                tD1[7:0] = cache_way2[set][offset];
                tD1[15:8] = cache_way2[set][offset + 1];
                tC1 = 7;
                @(posedge clk);
                tD1[7:0] = cache_way2[set][offset + 2];
                tD1[15:8] = cache_way2[set][offset + 3];
                read = 1;
            end
            if (mC1 == 4) begin
                read = 0;
                cache_way_tag2[set][11] = 0;
                cache_old2[set] = 0;
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 5) begin
                cache_way2[set][offset] = D1[7:0];
                read = 0;
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
            if (mC1 == 6) begin
                cache_way2[set][offset] = D1[7:0];
                cache_way2[set][offset + 1] = D1[15:8];
                read = 0;
                tC1 = 7;
                @(posedge clk);
                read = 1;
            end
        end
    end
```

```
    end

    assign D1 = (read == 0)? tD1: 16'bzzzzzzzzzzzzzzzz;
    assign C1 = (read == 0)? tC1: 3'bzzz;
    assign A2 = tA2;
    assign D2 = (write == 1)? tD2: 16'bzzzzzzzzzzzzzzzz;
    assign C2 = (write == 1)? tC2: 2'bzz;
endmodule


module CPU(output[14:0] A1, inout[15:0] D1, inout[2:0] C1, input clk, input
read, input[31:0] cache_hits);
    reg[14:0] tA1;
    reg[15:0] tD1;
    reg[2:0] tC1;
    integer y = 0;
    integer x = 0;
    integer k = 0;
    reg[18:0] pa = 0;
    reg[18:0] pc = `M * `K + `K * `N * 2;
    reg[18:0] pb = 0;
    reg[31:0] sum = 0;
    reg[31:0] isum = 1;
    integer calcer = 0;

    initial begin
        $display("hi :)");
        calcer += 1; // int8 *pa = a
        calcer += 1; // int32 *pc = c
        calcer += 1; // int y = 0
        for (y = 0; y < `M; y += 1) begin
            calcer += 2; // y += 1 and for
            calcer += 1; // int x = 0
            for (x = 0; x < `N; x += 1) begin
                calcer += 2; // x += 1 and for
                calcer += 1; // int16 *pb = b
                pb = `M * `K;
                sum = 0;
                calcer += 1; // int32 s = 0
                calcer += 1; // int k = 0
                for (k = 0; k < `K; k += 1) begin
```

```
                calcer += 2; // x += 1 and for
                calcer += 5 + 1; // mul and add
                calcer += 1; // add
                pa += k;
                isum = 1;
                tA1 = pa[18:4];
                tC1 = 1;
                @(posedge clk);
                @(posedge clk);
                tA1 = pa[3:0];
                while (C1 != 7) @(posedge clk);
                pb += x * 2;
                tA1 = pb[18:4];
                tC1 = 2;
                isum *= D1;
                @(posedge clk);
                @(posedge clk);
                tA1 = pb[3:0];
                @(posedge clk);
                while (C1 != 7) @(posedge clk);
                tC1 = 0;
                isum *= D1;
                pa -= k;
                pb -= x * 2;
                pb += `N * 2;
                sum += isum;
                while (C1 != 0)@(posedge clk);
            end
            pc += 4 * x;
            tA1 = pc[18:4];
            tD1 = sum[31:16];
            @(posedge clk);
            // $display("IMPORTANT: %d C1: %d pc: %d", A1, C1, pc);
            @(posedge clk);
            tC1 = 7;
            @(posedge clk);
            @(posedge clk);
            @(posedge clk);
            // $display("IMPORTANT: %d C1: %d pc: %d", A1, C1, pc);
            tA1 = pc[3:0];
            while (C1 != 0) @(posedge clk);
```

```
                while (C1 != 7) @(posedge clk);
                tD1 = sum[15:0];
                tC1 = 0;
                pc -= 4 * x;
                while (C1 != 0) @(posedge clk);
                $display("y: %d x: %d hits: %d", y, x, cache_hits);
            end
            pa += `K;
            calcer += 1; // add
            pc += `N * 4;
            calcer += 1; // add
        end
        $display("y: %d x: %d cache_hits: %d calcer: %d", y, x, cache_hits,
calcer);
        $finish;
    end

    always @(posedge clk) begin
        calcer += 1;
    end

    assign A1 = tA1;
    assign D1 = (read == 1)? tD1: 16'bzzzzzzzzzzzzzzzz;
    assign C1 = (read == 1)? tC1: 3'bzzz;
endmodule


module collect;
    reg[14:0] A1;
    wire[14:0] A2;
    wire[15:0] D1;
    wire[15:0] D2;
    wire[2:0] C1;
    wire[1:0] C2;
    reg[31:0] cache_hits;
    reg clk = 0;
    reg m_dump, reset, c_dump;
    wire write, read;
    integer i = 0;
```

```
    mem memCTR(.A2(A2), .D2(D2), .C2(C2), .clk(clk), .m_dump(m_dump),
.reset(reset), .write(write));
    CPU cpu(.A1(A1), .D1(D1), .C1(C1), .read(read), .clk(clk),
.cache_hits(cache_hits));
    cache mcache(.A1(A1), .A2(A2), .D1(D1), .D2(D2), .C1(C1), .C2(C2),
.c_dump(c_dump), .reset(reset), .write(write), .read(read),
.cache_hits(cache_hits), .clk(clk));
    initial begin
        // c_dump = 0;
        // m_dump = 0;
        // for (i = 0; i < 600; i += 1) begin
        //     @(posedge clk);
        // end
        // c_dump = 1;
        // m_dump = 1;
    end

    always #10 clk = ~clk;
endmodule
```