Лабораторная работа №2	M3136	2022
Моделирование схем в Verilog	Панюхин Никита Константинович	

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

Инструментарий и требования к работе: используемый язык: SystemVerilog, компиляция и симуляция – Icarus Verilog 12

Описание

<Пишем условие работы TODO Формулировка задачи из условия.>

Вариант: 1

Вычисление недостающих параметров системы

Параметры из условия для варианта №1:

- Ассоциативность: **CACHE_WAY = 2**
- Размер тэга адреса: **CACHE_TAG_SIZE = 10 бит**
- Размер кэш-линии (полезных данных): **CACHE_LINE_SIZE = 16 байт**
- Кол-во кэш-линий: **CACHE_LINE_COUNT = 64**
- Размер памяти: **MEM_SIZE = 512 Кбайт**

Вычислим остальные параметры:

- Размер кэша:

CACHE_SIZE = CACHE_LINE_SIZE × CACHE_LINE_COUNT = 64 × 16 байт = 1024 байт

- Кол-во наборов кэш-линий:

CACHE_SETS_COUNT = CACHE_LINE_COUNT / CACHE_WAY = 64 / 2 = 32

- Размер индекса в наборе кэш-линий:

$$\text{CACHE\_SET\_SIZE} = log_2(\text{CACHE\_SETS\_COUNT}) = log_2(32) = 5 \text{ бит}$$

- Размер смещения:

$$\text{CACHE\_OFFSET\_SIZE} = log_2(\text{CACHE\_LINE\_SIZE}) = log_2(16 \text{ байт}) = 4 \text{ бит}$$

- Размер адреса:

$$\text{CACHE\_ADDR\_SIZE} = log_2(\text{MEM\_SIZE}) = log_2(512 \cdot 1024 \text{ байт}) = 19 \text{ бит}$$

Размер частей адреса кэша:

	tag	set	offset
Размер	CACHE_TAG_SIZE	CACHE_SET_SIZE	CACHE_OFFSET_SIZE
	10 бит	5 бит	4 бита
	19 бит		

Таблица №1 – Интерпретация адреса кэшем

Отсюда получим размеры шин A1 и A2:

- $\text{ADDR1\_BUS\_SIZE} = max(\text{CACHE\_TAG\_SIZE} + \text{CACHE\_SET\_SIZE}, \text{CACHE\_OFFSET\_SIZE}) = max(10 + 5, 7) = 15 \text{ бит}$
- **ADDR2_BUS_SIZE = CACHE_TAG_SIZE + CACHE_SET_SIZE = 15 бит**

Размеры шин C1 и C2, очевидно, равны логарифму от максимального значения сигнала на них, то есть $log_2(7 + 1) = 3$ и $log_2(3 + 1) = 2$ соответственно.

Все параметры системы указаны в файле src/parameters.sv

Аналитическое решение задачи

Аналитическое решение было выполнено с помощью кода на языке Python (находится в файле “`Analytical solution/solution.py`”).

Поскольку в задаче требуется лишь посчитать количество кэш-попаданий и суммарное число тактов работы программы, можно смоделировать лишь часть процессов, достаточную для получения ответа. Так, например, можно опустить все операции с памятью и значениями переменных, поскольку, записанные в память (Mem) не влияют на оба требуемых ответа. Также, поскольку в задаче нет вызовов операции `read32`, упрощается операция `read_bytes` – байты данных передаются одновременно в такт с командой `C1_RESPONSE`, и считать дополнительный такт не надо[→].

В остальном, программа на питоне полностью повторяет программу на Verilog, смысл большинства задержек подписан в коде. В некоторых местах требуется подождать изменения значения синхронизации (например, команда отправляется по спаду `CLK` и принимается по фронту `CLK`, см. пункт Соглашения), для этого используется функция `Cache.wait_clk`

В результате получаем следующие ответы:

```
Total time: 4613397.0 tacts
Cache hits: 228080/249600 = 91.38%
```

Моделирование заданной системы на Verilog

Поскольку система в чём-то вариативна, нужно описать некоторые соглашения. Опишу пунктами

При моделировании были использованы следующие соглашения:

Далее соглашения будут помечены значком «➢»

Общение по шине:

- ➢ Данные на шине меняются при `CLK = 0` (по спаду, `negedge`), а принимаются на другой стороне при `CLK = 1` (по фронту, `posedge`)

// Примеры:

// байт(bbytes_start + bbyte) -> D2[pos_left:pos_right]

// 0 + 0 -> [7:0]

// 0 + 1 -> [15:8]

// 1 + 0 -> [7:0]

// 1 + 1 -> [15:8]

// 2 + 0 -> [7:0]

// 2 + 1 -> [15:8]

// Тогда: байт [bbytes_start + bbyte] нужно записать в D2[bbyte * 8 + 7:bbyte * 8]

// Пример: отправялем два байта reg byte1 = 236 (11101100); reg byte2 = 122 (01111010)

// Тогда отправится reg D2 = 01111010|11101100 - конкатенация двух байтов

NOP ставиться по спаду CLK, когда он равен 0

CPU встроен в testbench

```
[424 | CLK=0] MemCTR: Sent byte  31 = 00011111 from ram[00000000010001000011]
[424 | CLK=0] C1_WIRE = 0, C2_WIRE = 1
[425 | CLK=1] Cache: Wrote byte  50 = 00110010 to data[2][1][10]
[425 | CLK=1] Cache: Wrote byte  31 = 00011111 to data[2][1][11]
[425 | CLK=1] C1_WIRE = 0, C2_WIRE = 1
[426 | CLK=0] MemCTR: Sent byte 245 = 11110101 from ram[00000000010001000100]
[426 | CLK=0] MemCTR: Sent byte 181 = 10110101 from ram[00000000010001000101]
[426 | CLK=0] C1_WIRE = 0, C2_WIRE = 1
[427 | CLK=1] Cache: Wrote byte 245 = 11110101 to data[2][1][12]
[427 | CLK=1] Cache: Wrote byte 181 = 10110101 to data[2][1][13]
[427 | CLK=1] C1_WIRE = 0, C2_WIRE = 1
[428 | CLK=0] MemCTR: Sent byte  35 = 00100011 from ram[00000000010001000110]
[428 | CLK=0] MemCTR: Sent byte 231 = 11100111 from ram[00000000010001000111]
[428 | CLK=0] C1_WIRE = 0, C2_WIRE = 1
[429 | CLK=1] Cache: Wrote byte  35 = 00100011 to data[2][1][14]
[429 | CLK=1] Cache: Wrote byte 231 = 11100111 to data[2][1][15]
[429 | CLK=1] C1_WIRE = 0, C2_WIRE = 1
[430 | CLK=0] Cache: Wrote byte 124 = 01111100 to data[2][1][3]
[430 | CLK=0] Cache: Wrote byte 200 = 11001000 to data[2][1][4]
[430 | CLK=0] Cache: Wrote byte   5 = 00000101 to data[2][1][5]
[430 | CLK=0] Cache: Wrote byte  37 = 00100101 to data[2][1][6]
[430 | CLK=0] C1_WIRE = 7, C2_WIRE = z
[431 | CLK=1] C1_WIRE = 7, C2_WIRE = z
[431 | CLK=1] CPU received C1_RESPONSE

---------- Iteration 0 finished ----------
```

После read_line_from_Mem мы ждём лишний такт. Можно без него.

Переход на следующую итерацию цикла +#2 лишних оставим

Воспроизведение задачи на Verilog

Сравнение полученных результатов

Листинг кода