

Симуляция, управляемая событиями

Курс «Программное моделирование вычислительных систем»

Григорий Речистов
grigory.rechistov@phystech.edu

22 января 2015 г.

- 1 Таймер
- 2 Отложенный ответ
- 3 Теория
- 4 Практический пример
- 5 Косимуляция
- 6 Практический пример
- 7 Заключение
- 8 Литература
- 9 Конец

Пример №1: таймер

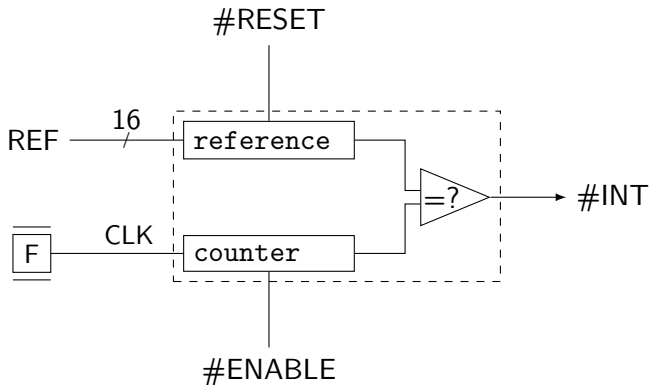
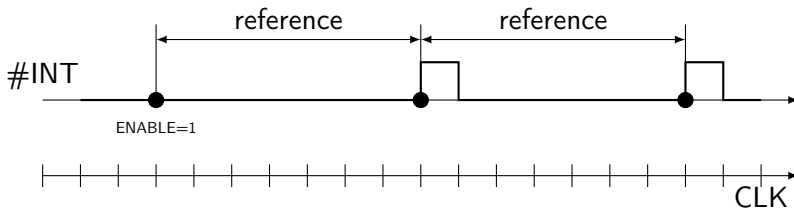


Диаграмма работы



Моделирование с фиксированным шагом

```
on_clk() {  
    if (enable) counter +=1;  
    if (counter == reference) {  
        raise_int();  
        counter = 0;  
    } else {  
        lower_int();  
    }  
}
```

```
on_reset() {  
    reference = 0;  
    counter = 0;  
}
```

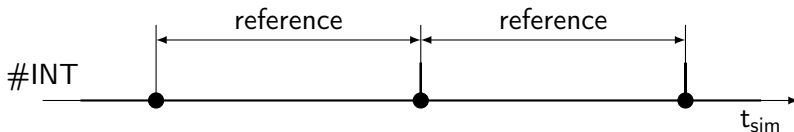
Типичные значения параметров таймера

- $F \approx 10 \text{ МГц}$
- $\text{reference} > 10^3$
- $\# \text{RESET}$ — не чаще одного раза в ≈ 100 секунд

⇒ внешне видимый эффект ($\# \text{INT}$) происходит примерно один раз в 10^3 тактов.

Оптимизация

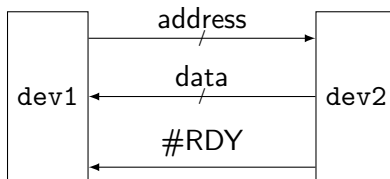
Не моделировать внешне ненаблюдаемые действия.



Моделирование событий

```
struct event_t {  
    time_t delta;  
    dev_t *device;  
    (*function)();  
}  
  
event_t event_queue[100];  
time = 0;  
foreach e in event_queue {  
    e.function(e.device);  
    time +=e.delta;  
}
```


Пример №2: ожидание ответа



- 1 Запрос от dev1: address.
- 2 dev2 вычисляет data.
- 3 dev2 оповещает dev1 о готовности данных через некоторое время ΔT с помощью #RDY.
- 4 dev1 после отправки address и до получения #RDY работает независимо.

Реализация

dev1:

- 1 dev2.read(address)

dev2:

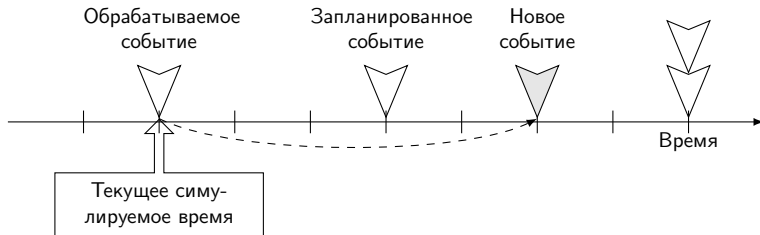
- 1 data = get_data(address)

- 2 event_queue.post(ΔT , dev1, rdy())

dev1:

- 1 rdy(): чтение data.

Очередь событий



Содержимое и результаты

Что содержится в одном событии:

- функция, которая должна быть вызвана,
- объект, состояние которого изменяется.

Результаты обработки события:

- изменение состояния системы,
- добавление/уничтожение событий.

Вопросы

Что будет происходить с очередью событий при

- 1 записи в `reference` или `#RESET`?
- 2 выключении таймера ($\text{ENABLE} \leftarrow 0$)?
- 3 чтении регистра `counter`?

Алгоритм DES

```
struct event_t {  
    time_t delta;  
    dev_t *device;  
    (*function)();  
}  
  
uint sim_time = 0;  
while (! empty(queue)) {  
    sim_time += get_delta(queue);  
    evt_t evt = pop(queue);  
    evt.fn(evt.device, queue);  
}
```

Пример на модели or1k

Демо

```
simics> log-level 1
```

```
New global log level: 1
```

```
simics> continue-cycles 199
```

```
[chip0] v:0x031c p:0x031c  nop
```

```
simics> peq
```

Step	Object	Description
Cycle	Object	Description
1	tick0	reference_reached
499802	cosim_cell	sync_report
999801	sim	Time Quantum End
999801	cosim_cell	sync_block

На этой лекции

- Совместная работа с моделью процессора.
- Работа с несколькими процессорами сразу.

Какие типы моделей нам известны

- Интерпретация

Какие типы моделей нам известны

- Интерпретация: процессоры.

Какие типы моделей нам известны

- Интерпретация: процессоры.
- Очередь событий

Какие типы моделей нам известны

- Интерпретация: процессоры.
- Очередь событий: таймер.

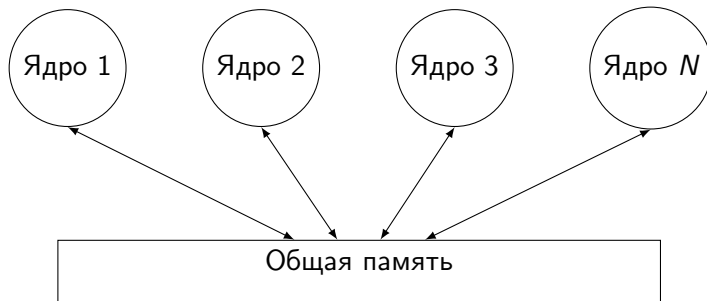
Какие типы моделей нам известны

- Интерпретация: процессоры.
- Очередь событий: таймер.
- «Мгновенная» модель «стимул — отклик»

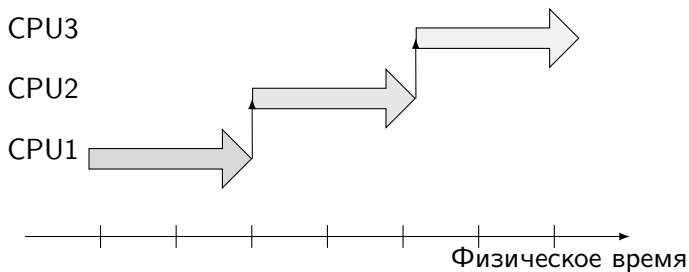
Какие типы моделей нам известны

- Интерпретация: процессоры.
- Очередь событий: таймер.
- «Мгновенная» модель «стимул — отклик»: ОЗУ.

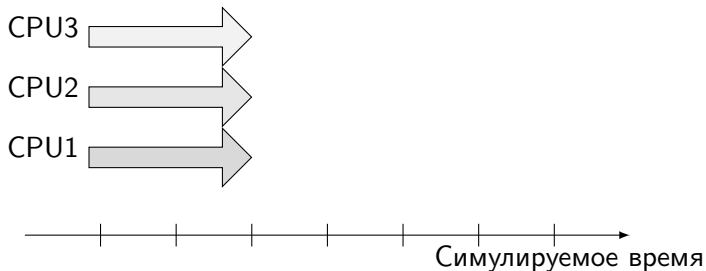
Модель многопроцессорной системы



Квотированная симуляция



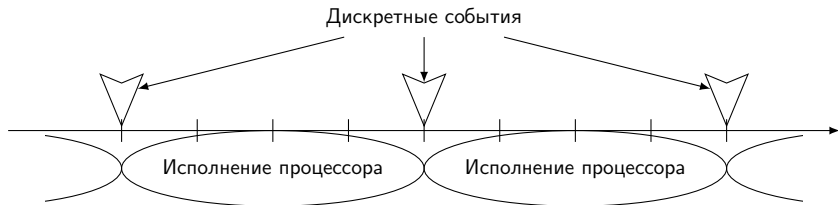
Квотированная симуляция



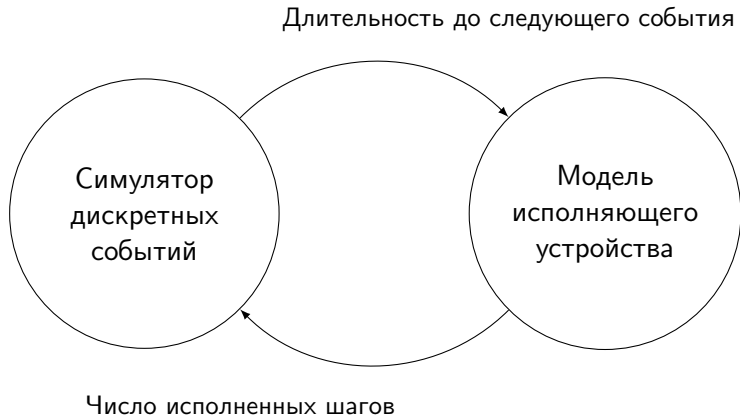
Размер квоты

- Процессор может исполнить меньше инструкций, чем содержится в выданной ему квоте.
- Не следует увлекаться излишне большими квотами, пытаясь ускорить исполнение.
- В DES-модели могут быть реализованы псевдо-события, обработка которых вызывает переключение текущего процессора.

Совместная симуляция DES и исполняющей модели



Косимуляция



Пример на модели viper-busybox.simics

Квота на восьмиядерной конфигурации:

```
simics> cpu-switch-time
Current time quantum:      0.0001 s
  200000.0  viper.mb.cpu0.core[0][0]
  200000.0  viper.mb.cpu0.core[0][1]
  200000.0  viper.mb.cpu0.core[1][0]
  200000.0  viper.mb.cpu0.core[1][1]
  200000.0  viper.mb.cpu0.core[2][0]
  200000.0  viper.mb.cpu0.core[2][1]
  200000.0  viper.mb.cpu0.core[3][0]
  200000.0  viper.mb.cpu0.core[3][1]
Default time quantum not set yet
```

Пример на модели viper-busybox.simics

Симулируемое время на восьмиядерной конфигурации:

```
simics> ptime -all
```

processor	steps	cycles	time [s]
viper.mb.cpu0.core[0][0]	4602030000	5134030000	2.567
viper.mb.cpu0.core[0][1]	4925600000	5134000000	2.567
viper.mb.cpu0.core[1][0]	4925600000	5134000000	2.567
viper.mb.cpu0.core[1][1]	4925600000	5134000000	2.567
viper.mb.cpu0.core[2][0]	4925600000	5134000000	2.567
viper.mb.cpu0.core[2][1]	4925600000	5134000000	2.567
viper.mb.cpu0.core[3][0]	4925600000	5134000000	2.567
viper.mb.cpu0.core[3][1]	4925600000	5134000000	2.567

Дополнительные вопросы DES

- Совместная работа с моделью процессора.
- Работа с несколькими процессорами сразу.
- Сценарии, когда действительно надо моделировать каждый такт.

Литература I



Паттерсон Д., Хеннесси Дж. Архитектура компьютеров и проектирование компьютерных систем — СПб.: Питер — 2011 — 784 с. — ISBN 978-5-459-00291-1



Handbook of Simulation. Principles, Methodology, Advances, Applications, and Practice / ed. by J. Banks. — John Wiley & Sons, Inc., 1998. — ISBN 0-471-13403-1. — <http://books.google.com/books?id=dMZ1Zj3TBgAC>

Спасибо за внимание!

Слайды и материалы курса доступны по адресу

<http://is.gd/ivuboc>

Замечание: все торговые марки и логотипы, использованные в данном материале, являются собственностью их владельцев. Представленная точка зрения отражает личное мнение автора. Материалы доступны по лицензии Creative Commons Attribution-ShareAlike (Атрибуция — С сохранением условий) 4.0 весь мир (в т.ч. Россия и др.). Чтобы ознакомиться с экземпляром этой лицензии, посетите

<http://creativecommons.org/licenses/by-sa/4.0/>