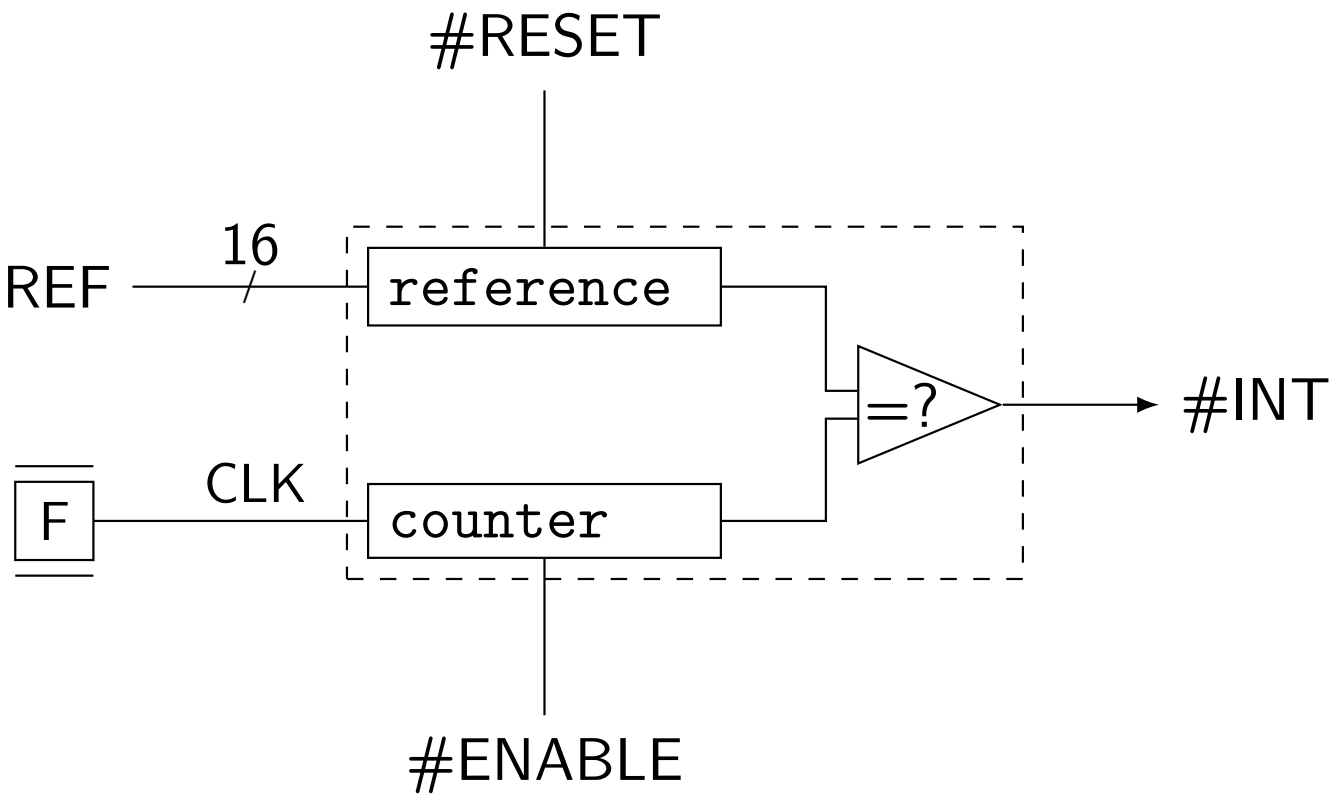
# Симуляция, управляемая событиями. Многоагентная симуляция

## Курс «Программное моделирование вычислительных систем»

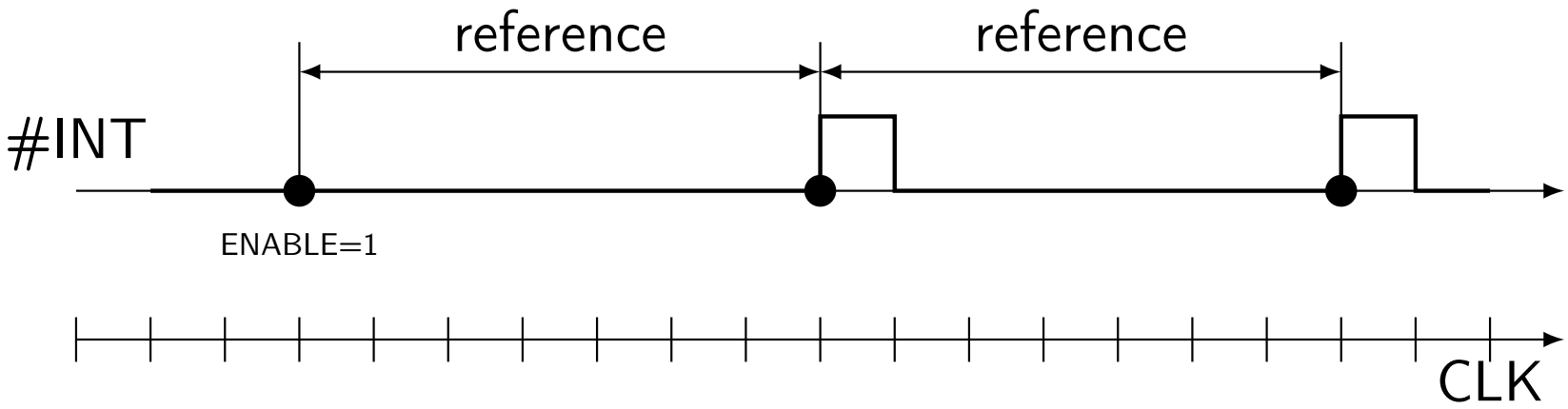
Григорий Речистов
grigory.rechistov@phystech.edu

13 сентября 2014 г.

# 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;
}
```

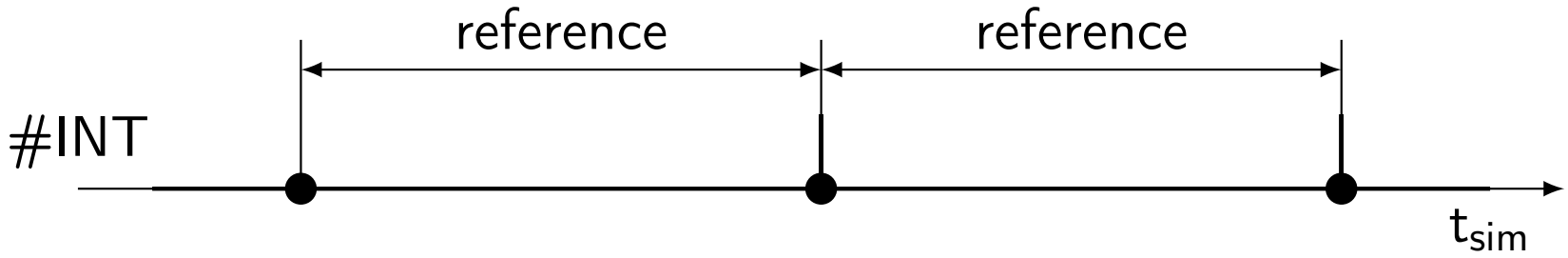
# Типичные значения параметров таймера

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

$\Rightarrow$ внешне видимый эффект (#INT) происходит примерно один раз в $10^3$ тактов.

# Оптимизация

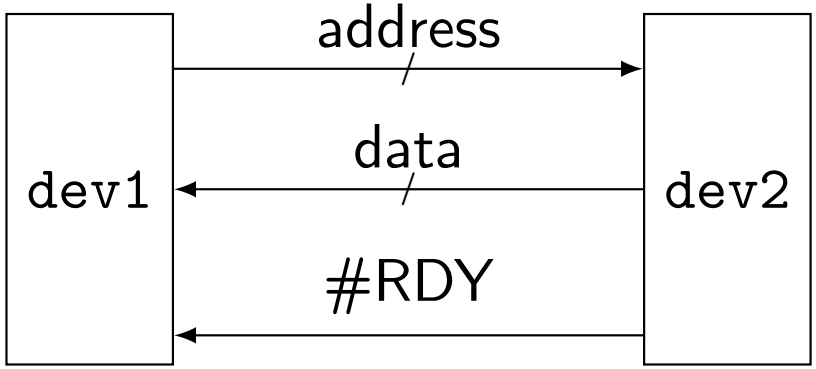
Не моделировать внешне ненаблюдаемые действия.



# 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* $\Delta T$ 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($\Delta T$, dev1, rdy())

`dev1`:

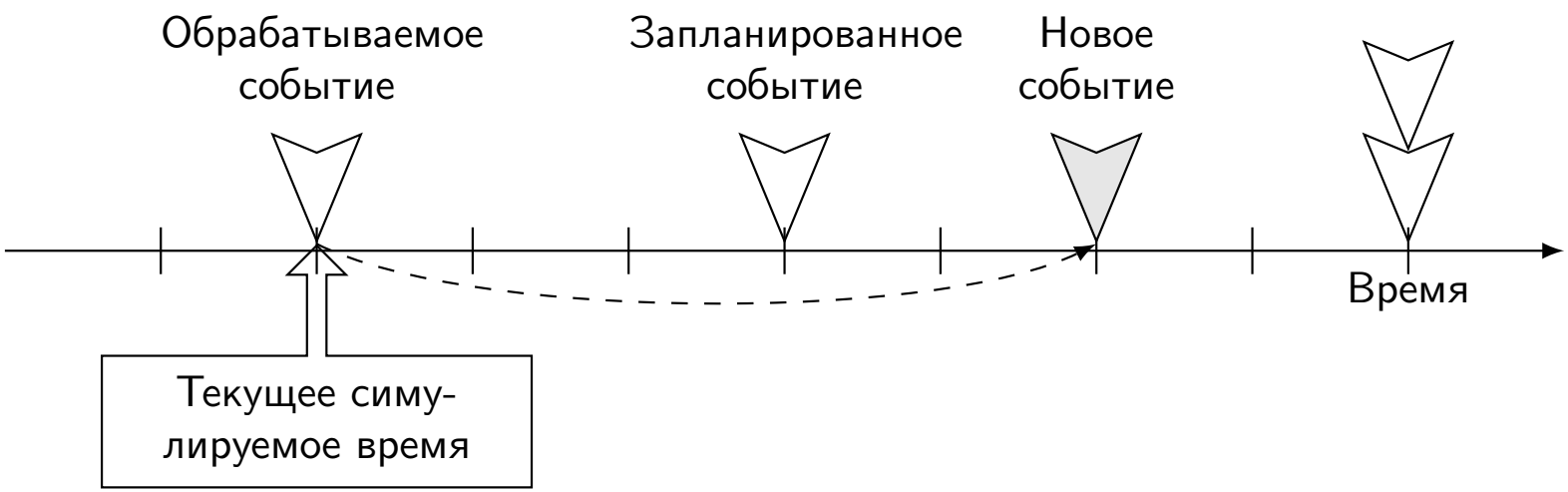
1. rdy(): чтение data.

# Утрясаем детали

Что будет происходить с очередью событий при

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

# Очередь событий



# Что содержится в одном событии

- Функция, которая должна быть вызвана.
- Объект, состояние которого изменяется.

Результаты обработки события

- Изменение состояния системы.
- Добавление/уничтожение событий.

# 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);
}
```

# Свойства событий

- Порождаемые события не могут попасть в прошлое.
- Обработка событий может не только порождать события в будущем, но и отменять некоторые из них (ещё не обработанные).
- Несколько событий могут иметь одинаковую метку времени.

# Пример на модели or1k

```
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
```

# Что осталось нерассказанным

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

# Литература I

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