Языки описания схем

(mk.cs.msu.ru → Лекционные курсы → Языки описания схем)

Блок 21

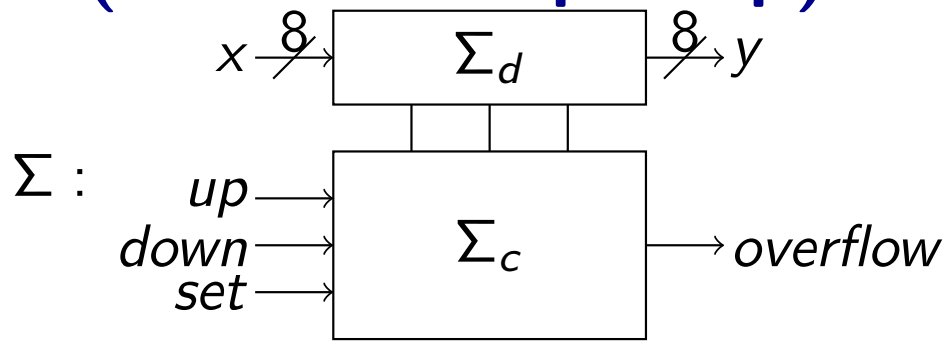
Как спроектировать
операционный автомат

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2019, осенний семестр

Напоминание (сквозной пример)



$y(0) = 0$

$overflow(0) = 0$

Если $set(t) = 1$, то $y(t) = x(t)$

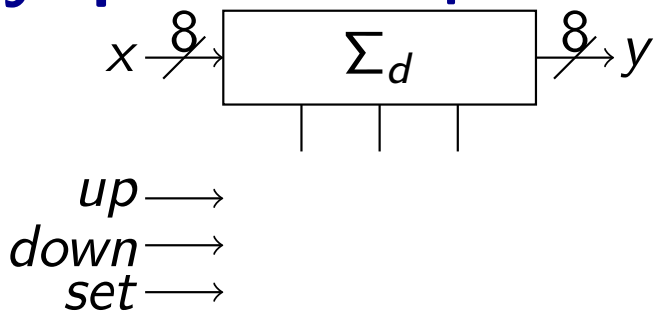
Если $set(t) = 0$ и $up(t) = 1$, то $y(t) = y(t-1) + 1$ (*по модулю* 2^8)

Если $set(t) = up(t) = 0$ и $down(t) = 1$, то $y(t) = y(t-1) - 1$

Иначе $y(t) = y(t-1)$

Если при переходе от $y(t-1)$ к $y(t)$ произошло арифметическое переполнение, то $overflow(t) = 1$, а иначе $overflow(t) = 0$

Забываем про управляющий автомат



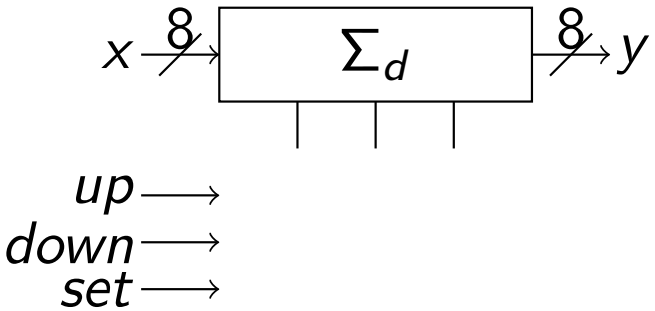
Забудем на время про

- подсхему Σ_c и
- точный смысл портов *up*, *down*, *set*, *overflow*, и

попробуем реализовать *операционный автомат* Σ_b, помня *в общих чертах*, какой смысл имеют входные управляющие порты

Для этого зададимся таким вопросом:
что в принципе должна уметь делать схема
(с данными x и не только с ними),
чтобы всегда можно было предоставить требуемые данные y?

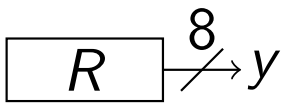
Проектируем память



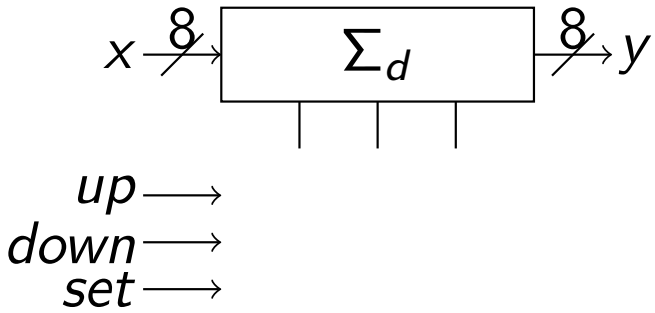
1. Определимся с основными ячейками памяти операционного автомата

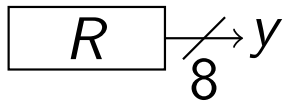
Данные y **не**однозначно определяются значениями на входах — значит, их потребуется извлекать из триггеров/регистров

Пойдём по простому пути:
y — **состояние параллельного регистра** R



Определяем вспомогательные подсхемы





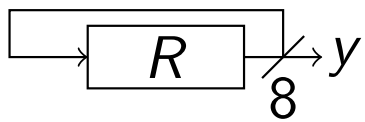
2. Для каждого способа преобразования данных (*независимо от остальных*) подумаем, что потребуется добавить в схему

Первый способ: $y(0) = 0$

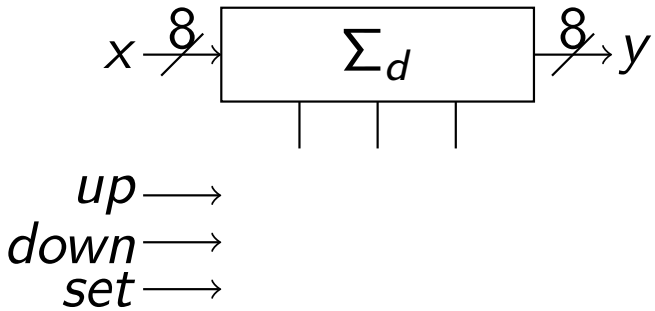
Значит, R — регистр со сбросом, и больше ничего не требуется

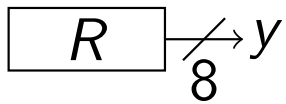
Второй способ: $y(t) = y(t-1)$

Достаточно направить выход R на вход:



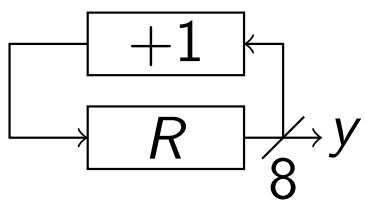
Определяем вспомогательные подсхемы



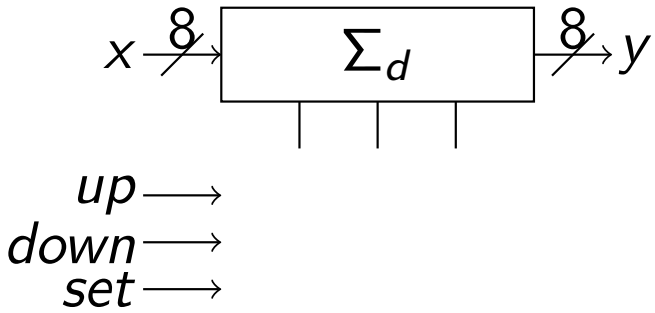


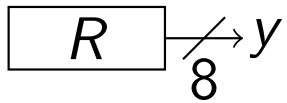
Третий способ: $y(t) = y(t-1) + 1$

Достаточно направить выход R на вход,
поставив посередине подходящую *комбинационную* схему:



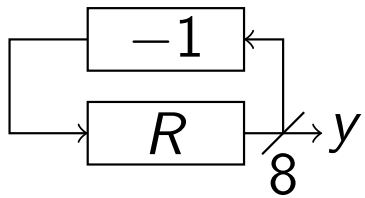
Определяем вспомогательные подсхемы





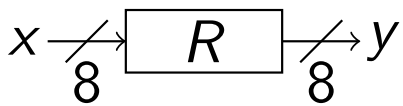
Четвёртый способ: $y(t) = y(t-1) - 1$

Здесь всё то же самое, только комбинационная схема другая:

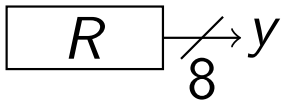


Пятый способ: $y(t) = x(t)$

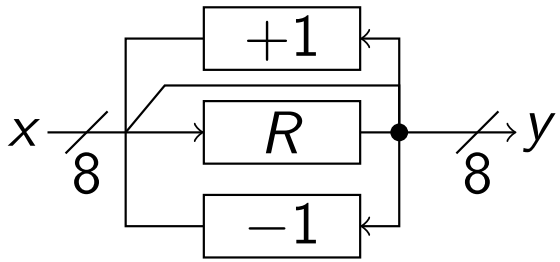
Достаточно направить x на вход R:



Совмещаем вспомогательные подсхемы



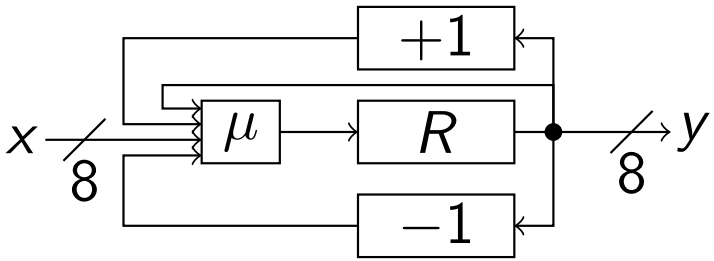
3. Совместим получившиеся схемы, “наложив” их друг на друга



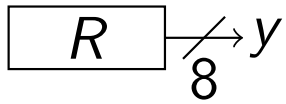
Кажется, что-то пошло не так ☹

При совмещении способов преобразования данных “в лоб” возникли **конфликты пересылки данных**: при разных обстоятельствах на вход R посылаются значения из разных точек схемы

Типовой способ разрешения таких конфликтов: добавим *мультиплексор* (μ) в точку конфликта, и пусть **управляющий автомат** решает, какое значение должно быть на *управляющем входе* μ



Итог



Примерно так обычно и разрабатывается операционный автомат:

1. Приблизительно расставляются триггеры/регистры данных
2. Всё то, **что** операционный автомат должен уметь делать с данными, покрывается подсхемами
3. В места возникновения конфликтов пересылки данных вставляются мультиплексоры (*и в операционном автомате появляются новые порты управляющих сигналов*)

Промежуточный итог для сквозного примера: