Языки описания схем

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

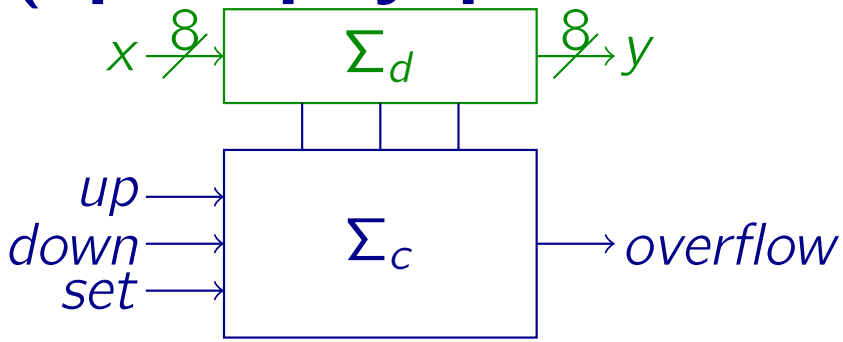
Блок 30

Как спроектировать
операционный автомат

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2025, сентябрь–декабрь

Напоминание (пример, управляемый счётчик)



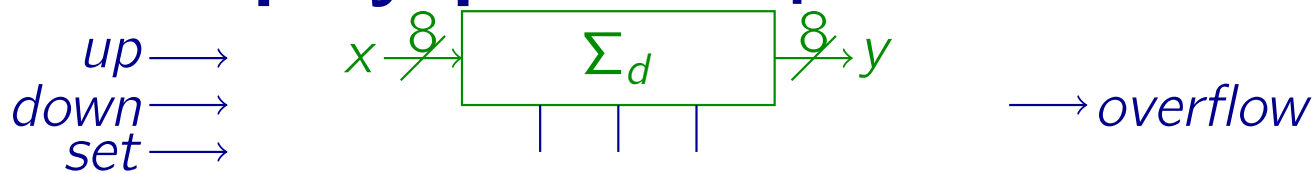
$$y(1) = 0 \qquad\qquad overflow(1) = 0$$

$$y(t+1) = \begin{cases} x(t), & \text{если } set(t) = 1; \\ y(t) + 1, & \text{если } set(t) = 0 \text{ и } up(t) = 1; \\ y(t) - 1, & \text{если } set(t) = up(t) = 0 \text{ и } down(t) = 1 \\ y(t) & \text{иначе} \end{cases}$$

$overflow(t+1) = 1 \quad \Leftrightarrow$
при переходе от $y(t)$ к $y(t+1)$ происходит арифметическое переполнение

На этом примере обсудим типовой способ разработки операционного автомата (Σ_d)

1. Забываем про управляющий автомат



Забудем на время про

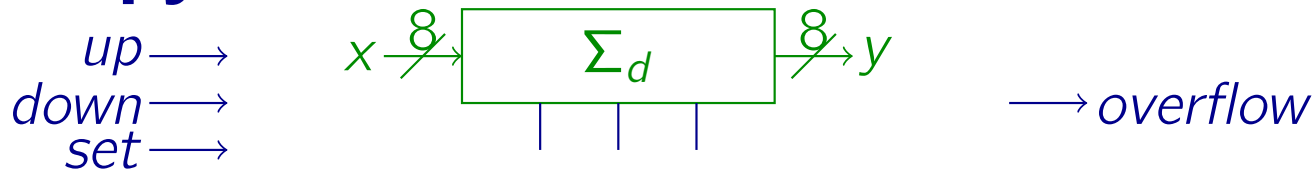
- управляющий автомат и
- *точную* зависимость значения $y(t+1)$ от значений $up(t)$, $down(t)$, $set(t)$, $overflow(t)$, и

попробуем реализовать операционный автомат, отталкиваясь **только** от спектра выражений, задающих $y(t+1)$

Для этого зададимся таким вопросом:

Что в целом должна уметь делать схема с данными (x и, быть может, не только), чтобы всегда была возможность получить требуемые выходные данные (y)?

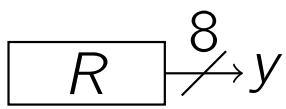
2. Проектируем память



Определимся с основными ячейками памяти операционного автомата

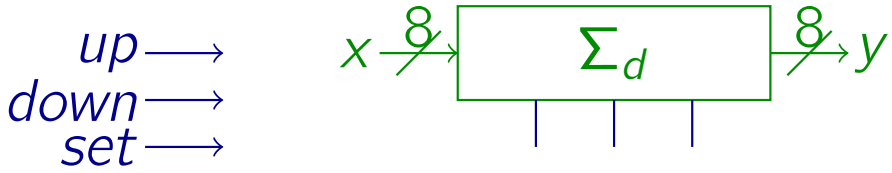
Данные y **не**однозначно определяются значениями на входах — значит, их потребуется извлекать из регистров

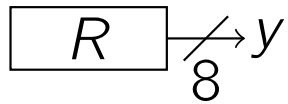
Пойдём по простому пути:
y — **состояние параллельного регистра (R)**



$y(1) = 0$, а значит, R — это регистром со сбросом

3. Определяем вспомогательные подсхемы

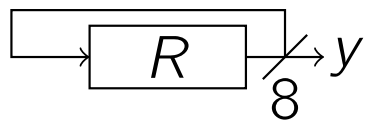




Для каждого способа преобразования данных
подумаем, что потребуется добавить в схему для этого преобразования

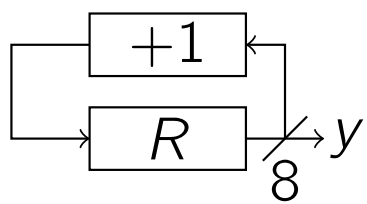
Первый способ: $y(t+1) = y(t)$

Достаточно направить выход R на вход:

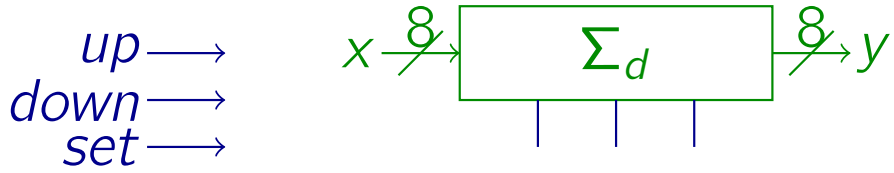


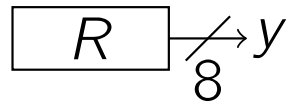
Второй способ: $y(t+1) = y(t) + 1$

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



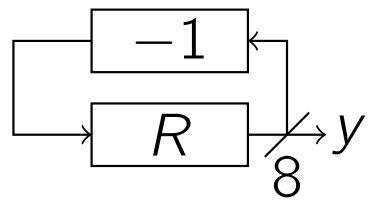
3. Определяем вспомогательные подсхемы





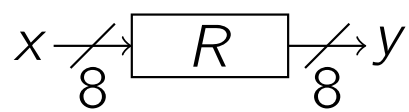
Третий способ: $y(t+1) = y(t) - 1$

Здесь всё то же самое, только комбинационная схема другая:



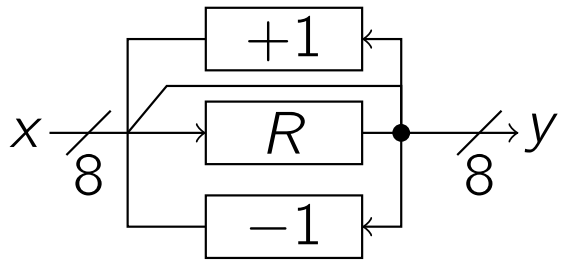
Четвёртый способ: $y(t+1) = x(t)$

Достаточно направить x на вход R:



4. Совмещаем вспомогательные подсхемы

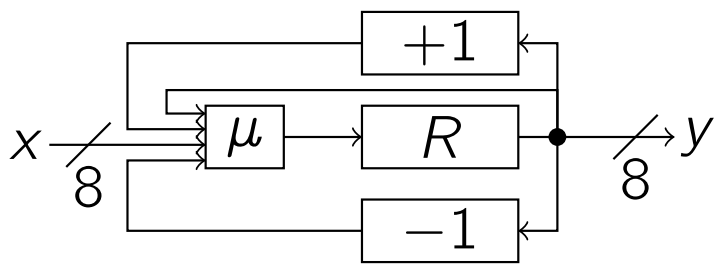
Совместим получившиеся схемы, «визуально наложив» их друг на друга



В схеме возникла точка конфликта пересылаемых данных, делающая схему некорректной: в одну точку (вход R) по задумке должно пересылаться несколько различных данных (x, y, $y+1$, $y-1$)

Но эти данные должны пересылаться не одновременно: в каждый момент времени должно выбираться только одно из этих значений

Типовой способ разрешения конфликта: добавим мультиплексор (μ) в точку конфликта, и оставим выбор значений управляющему автомату



Итог

Так обычно и разрабатывается операционный автомат:

1. Управляющий автомат откладывается в сторону
2. Приблизительно расставляются регистры данных
3. Реализуются все подсхемы преобразования данных
4. В точках конфликта расставляются мультиплексоры с добавлением соответствующих управляющих сигналов

Промежуточный итог для управляемого счётчика: