

Языки описания схем

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

Блок 31

Как дополнить
операционный автомат управляющим

Лектор:

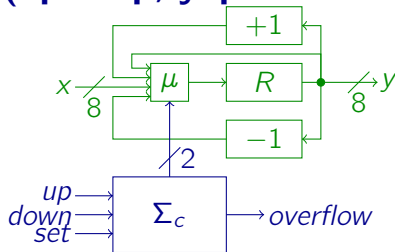
Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

ВМК МГУ, 2025, сентябрь–декабрь

Напоминание (пример, управляемый счётчик)



$$y(1) = 0$$

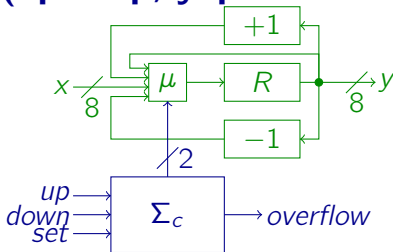
$$overflow(1) = 0$$

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

$$overflow(t+1) = 1 \Leftrightarrow$$

при переходе от $y(t)$ к $y(t+1)$ происходит арифметическое переполнение

Напоминание (пример, управляемый счётчик)



Попробуем, зная смысл всех управляющих сигналов и устройство операционного автомата, спроектировать **управляющий автомат Σ_c** . Для этого зададимся несколькими вопросами:

1. Как выходные значения Σ_c должны зависеть от входных, чтобы на выходах всей схемы появлялись подходящие значения?
2. Зависят ли выходные значения Σ_c от **данных**, содержащихся в **операционном автомате**?

При ответе на второй вопрос может понадобиться дополнить **операционный автомат** подсхемами, распознающими **свойства данных** для пересылки в **управляющий автомат**.

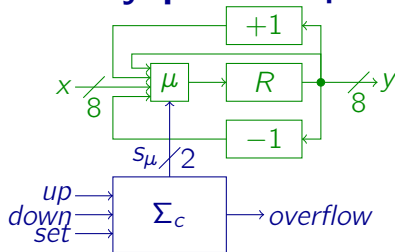
Типовое устройство управляющего автомата

Если текущее значение на выходе Σ_c **однозначно** определяется текущими значениями на входах, то входы и выход достаточно соединить подходящей комбинационной схемой

Иначе один из типовых способов разработки Σ_c — это разработка и типовая реализация подходящего **символьного автомата** $\mathcal{A}$:

- ▶ входные переменные $\mathcal{A}$ = входы Σ_c
- ▶ предикаты $\mathcal{A}$ = выражения выбранного языка описания схем
- ▶ выходные буквы $\mathcal{A}$ = наборы значений на выходах Σ_c
- ▶ входные буквы $\mathcal{A}$ читаются **при переходе от текущего такта к следующему в Σ_c**
- ▶ выходной символ в заданном состоянии $\mathcal{A}$ — это
 - ▶ значения на выходах Σ ,
которые должны **немедленно** установиться при переходе в состояние
 - ▶ значения, посылаемые в Σ_d и определяющие то, как состояние Σ_d
должно измениться **при переходе к следующему такту**

Как спроектировать управляющий автомат



Правда ли, что значение s_μ однозначно определяется значениями up , $down$ и set в тот же момент времени?

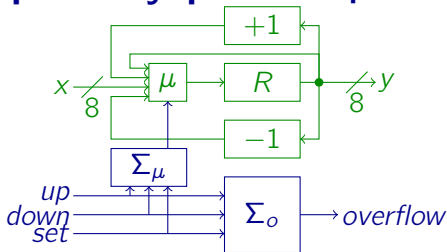
При переходе к следующему такту на выход μ должен перенаправляться

- ▶ *нижний* вход, если $set = 1$ (в тот же момент)
- ▶ *второй снизу* вход, если $set = up = 0$ и $down = 1$
- ▶ *второй сверху* вход, если $set = 0$ и $up = 1$
- ▶ *верхний* вход, если $up = down = set = 0$

Значит, ответ на поставленный только что вопрос — **да**

Обозначим соответствующую комбинационную схему записью Σ_μ

Как спроектировать управляющий автомат

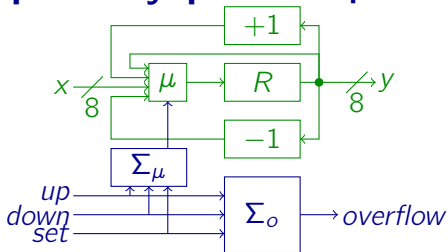


Правда ли, что значение *overflow* однозначно определяется значениями *up*, *down*, *set* в тот же момент времени?

Очевидно, что **нет**: текущее значение *overflow* зависит

- ▶ от значений *up*, *down*, *set* в момент перехода от предыдущего такта к текущему и
- ▶ от значения *y* на предыдущем такте

Как спроектировать управляющий автомат



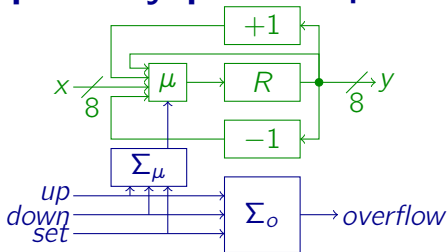
Проблема 1: *overflow* зависит от **данных** y

Если **строго** придерживаться принципа разделения данных и управления, то **запрещено** посылать **данные** y в Σ_o

Вместо этого можно

- ▶ выяснить, от каких *свойств* значения y зависит *overflow*
- ▶ добавить в **операционный автомат** подсхемы, распознающие эти свойства
- ▶ объявить результаты распознавания **управляющими сигналами** и послать в Σ_o

Как спроектировать управляющий автомат



Проблема 1: *overflow* зависит от **данных** y

Переполнение происходит в двух случаях:

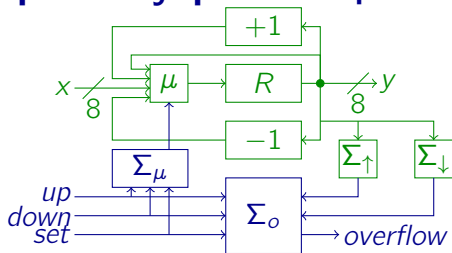
1. $y = (11111111)$ при увеличении значения
2. $y = (00000000)$ при уменьшении значения

Добавим в **операционный автомат** подсхемы $\Sigma_\uparrow$, $\Sigma_\downarrow$:

- ▶ обе принимают на вход y и выдают на выход булево значение ($b_\uparrow$ и $b_\downarrow$ соответственно)
- ▶ $b_\uparrow = 1 \Leftrightarrow y = (11111111)$
- ▶ $b_\downarrow = 1 \Leftrightarrow y = (00000000)$

Точки $b_\uparrow$, $b_\downarrow$ объявим **управляющими** и пошлём на вход Σ_o

Как спроектировать управляющий автомат

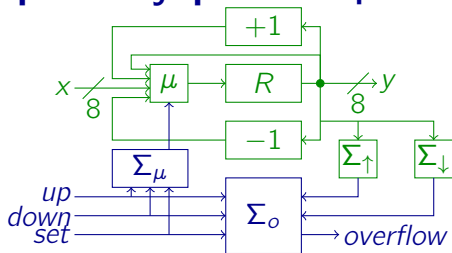


Проблема 2: *overflow* зависит от входов Σ_o **некомбинационно**

Попробуем придумать **символьный автомат** $\mathcal{A}_o$,
описывающий зависимость *overflow* от входов Σ_o

Если получится это сделать, то останется только
«бездумно» реализовать Σ_o как схему, соответствующую $\mathcal{A}_o$,
и схема Σ (наконец-таки!) будет полностью спроектирована

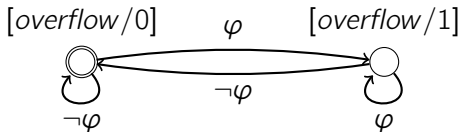
Как спроектировать управляющий автомат



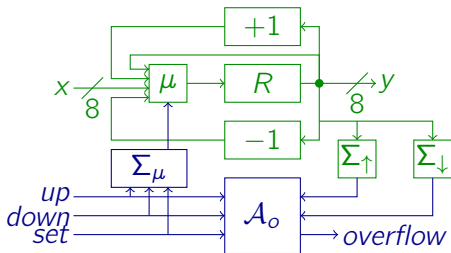
Проблема 2: *overflow* зависит от входов Σ_o **некомбинационно**

Ответ ($\mathcal{A}_o$):

$(\varphi: \neg \text{set} \ \& \ (up \ \& \ b_{\uparrow} \vee \neg up \ \& \ down \ \& \ b_{\downarrow}))$



Итог

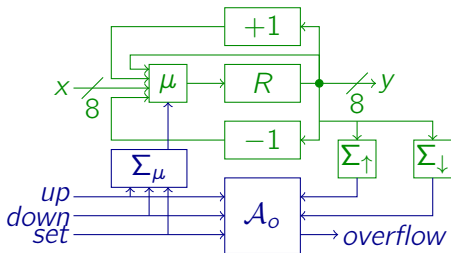


Разработка схемы **управляемого счётчика** со строгим разделением **данных** и **управления** завершена

В этом примере результат может показаться неоправданно громоздким и объёмным

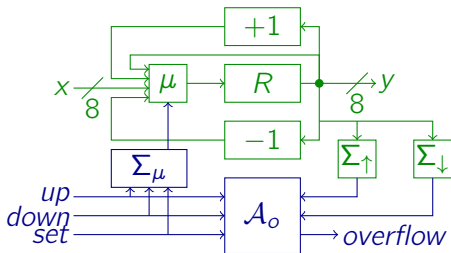
Это впечатление проистекает в основном из простоты примера и уравнивается массой преимуществ — в том числе: ...

Итог



- ▶ В этой схеме очень хорошо видно следование **модульной парадигме** программирования
- ▶ Поиск ошибки при отладке завершается в одной из небольших «логически целостных» подсхем, исправить ошибку достаточно просто, и исправление предсказуемо влияет на результат

Итог



- ▶ И разработка, и отладка схемы упрощаются тем, что проектировать каждую подсхему можно почти независимо от остальных, и для этого требуется размышлять ровно над одним из трёх вопросов, не смешивая их:
 - ▶ Какие способы преобразования данных нужны в схеме? (операционный автомат)
 - ▶ Как следует управлять этими способами? (управляющий автомат)
 - ▶ Какие свойства данных потребуются для такого управления? (реализация свойств данных в операционном автомате)