

Языки описания схем

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

Блок 22

Как дополнить
операционный автомат управляющим

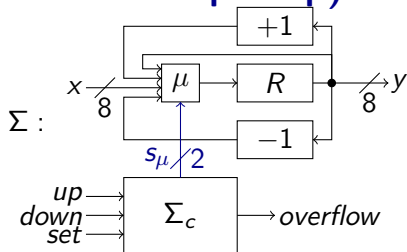
Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

Вступление (сквозной пример)



$$y(0) = 0$$

$$overflow(0) = 0$$

Если $set(t) = 1$, то $y(t) = x(t)$

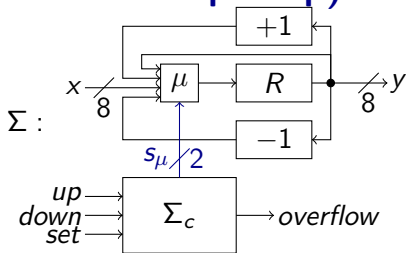
Если $set(t) = 0$ и $up(t) = 1$, то $y(t) = y(t-1) + 1$ (по модулю 2^8)

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 $overflow(t) = 0$

Вступление (сквозной пример)



Попробуем, зная смысл всех управляющих сигналов и устройство операционного автомата, спроектировать *управляющий автомат* Σ_c

Для этого зададимся несколькими вопросами:

Как значения выходных сигналов Σ_c зависят от входных?

Зависят ли выходные сигналы Σ_c от *данных*, содержащихся в операционном автомате (Σ_d)?

При ответе на второй вопрос может понадобиться дополнить операционный автомат подсхемами, собирающими сводную информацию о данных для пересылки в управляющий автомат

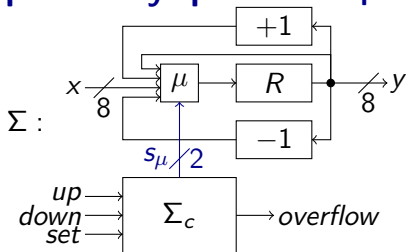
Типичное устройство управляющего автомата

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

Если предыдущее условие неверно, то (*нередко*) Σ_c — это 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
- ▶ в выходном символе состояния \mathcal{A} записаны
 - ▶ 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



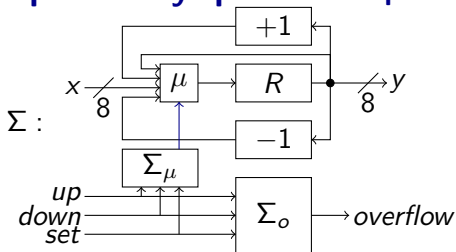
Правда ли, что значение s_μ однозначно определяется значениями up , $down$ и set в тот же момент времени?

На выход μ в момент чтения символа должен направляться

- ▶ *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
(явно рисовать эту схему не будем, обозначим её записью Σ_μ)

Как спроектировать управляющий автомат

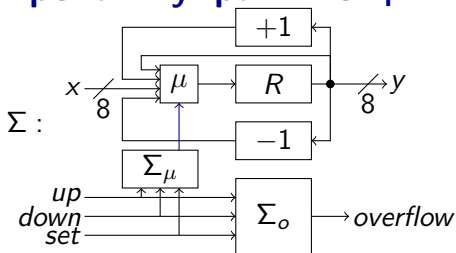


Правда ли, что значение *overflow* однозначно определяется значениями *up*, *down*, *set* в тот же момент времени?

Очевидно, что нет: текущее значение *overflow*

1. зависит и от значения y
2. зависит от значений up , $down$, set и y не в тот же момент, а в момент перехода от предыдущего такта к текущему

Как спроектировать управляющий автомат



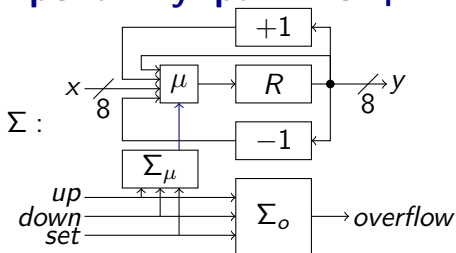
Проблема 1: *overflow* зависит от **данных** y

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:

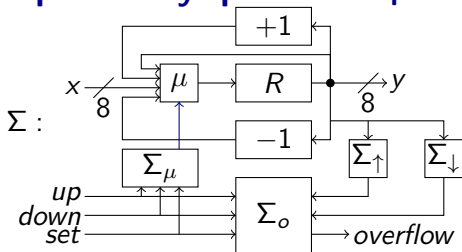
1. $y = (11111111)$, и значение y должно увеличиться
2. $y = (00000000)$, и значение y должно уменьшиться

Добавим в операционный автомат подсхемы Σ_\uparrow , Σ_\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

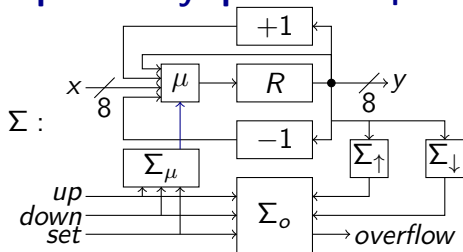


Проблема 2: *overflow* зависит от входов Σ_o некомбинационно

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

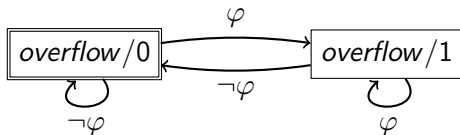
Как спроектировать управляющий автомат



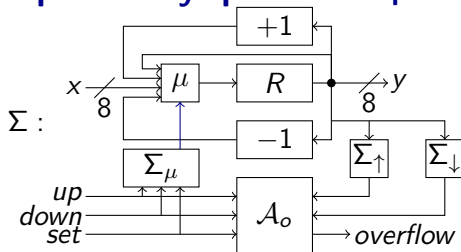
Проблема 2: *overflow* зависит от входов Σ_o некомбинационно

Ответ (\mathcal{A}_o):

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



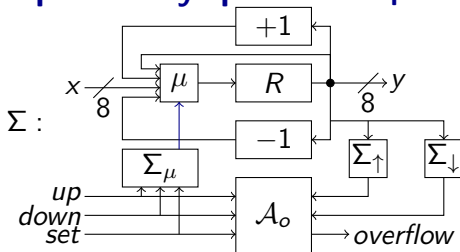
Как спроектировать управляющий автомат



Итог: получилась “грамотная” схема, которая

- ▶ на первый взгляд, выглядит страшно и непонятно
- ▶ если присмотреться, иллюстрирует **модульный подход** к программированию, адаптированный к схемной парадигме:
 - ▶ каждая деталь получившейся схемы решает маленькую “логически целостную” “независимую” подзадачу
 - ▶ замена каждой детали (если вдруг понадобится) требует малых трудозатрат и предсказуемо влияет на схему целиком

Как спроектировать управляющий автомат



Итог: получилась “грамотная” схема, которая

- ▶ аккуратно разделена на три части:
 - ▶ подсхема, принимающая все важные решения (управляющий автомат)
 - ▶ подсхема, выполняющаяся согласно этим решениям (операционный автомат до добавления подсхем-распознавателей)
 - ▶ подсхема, анализирующая свойства данных, требуемые для принятия решений (подсхемы-распознаватели операционного автомата)