

Языки описания схем

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

Блок 15

Как спроектировать операционный автомат
Комбинационный управляющий автомат

лектор:

Подымов Владислав Васильевич

e-mail:

valdus@yandex.ru

Осень 2018

Вступление

Проектирование операционного автомата имеет смысл только при соблюдении принципа разделения данных и управления

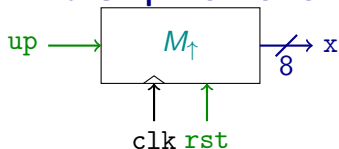
Какие сигналы/контакты/точки — данные, а какие управляют данными?

Откуда берутся данные, как и когда они изменяются и куда деваются?

Если не задумываться об оптимальности схемы, то структура операционного автомата очевидно определяется ответами на эти вопросы

Небольшое замечание: обычно единственная задача тактового сигнала — это синхронизация работы последовательных элементов схемы, поэтому он не будет особо упоминаться ни в данных, ни в управлении

Пример: увеличивающийся счётчик



После переднего фронта `rst` в `x` непрерывно выводится число 0

Выводимое число увеличивается на 1 по переднему фронту `clk`, если `up = 1`

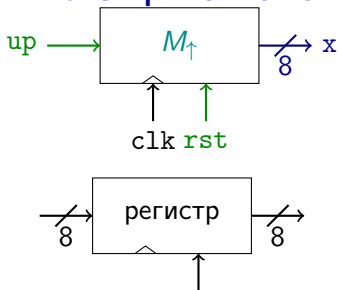
Какие здесь данные, и где управление?

“Полезная информация”, которую мы хотим получить от схемы — это число на выходе `x`

Значит, выход `x` относится к **данным**

Входы `up` и `rst` **управляют** тем, какое именно значение пересылается в `x`

Пример: увеличивающийся счётчик

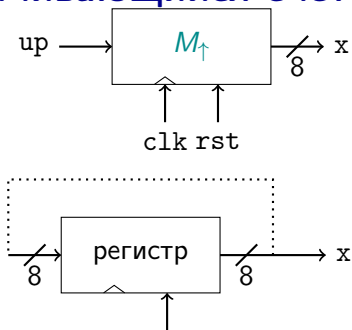


Откуда берутся данные?

Значение на выходе x изменяется по передним фронтам сигналов clk и rst , причём по фронту сигнала rst оно перезаписывается константным образом

Значит, разумно его хранить в параллельном регистре с асинхронным сбросом

Пример: увеличивающийся счётчик

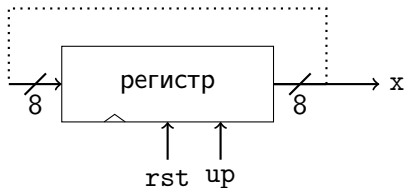
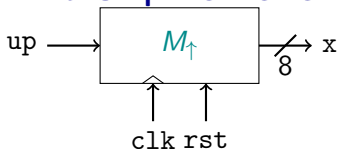


Куда деваются данные?

Значение, сохранённое в регистре,

- ▶ непрерывно передаётся на выход
- ▶ используется при изменении сохранённого значения

Пример: увеличивающийся счётчик



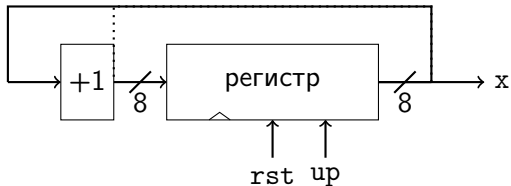
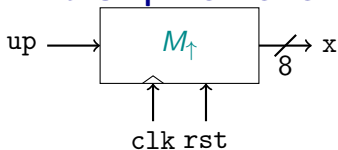
Когда изменяются данные?

Изменение данных по фронту сигнала rst учтено в схеме

Кроме того, данные изменяются по переднему фронту тактового сигнала, если $up = 1$, и не изменяются, если $up = 0$

Это типичное поведение регистра с синхронным включением загрузки

Пример: увеличивающийся счётчик

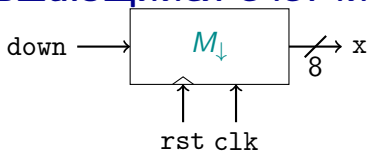


Как изменяются данные?

Если загрузка включена, то во время перезаписи данные увеличиваются на единицу

Значит, на вход регистра должно подаваться значение с выхода, к которому прибавлена единица

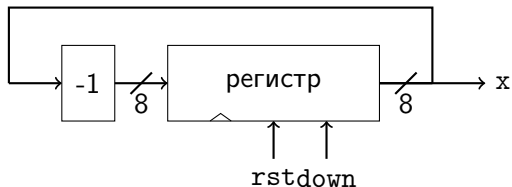
Пример: уменьшающийся счётчик



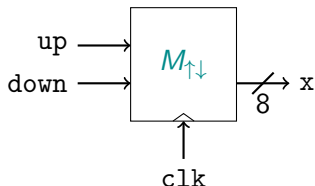
После переднего фронта `rst` в `x` непрерывно выводится число 0

Выводимое число уменьшается на 1 по переднему фронту `clk`, если `down = 1`

Аналогичными рассуждениями приходим к такому операционному автомату:



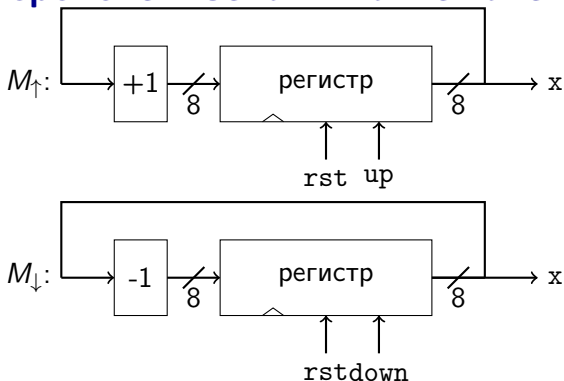
Пример переиспользования автоматов



В x непрерывно выводится последнее сохранённое число, изменяющееся по следующим правилам:

- ▶ всегда, когда $up = down = 1$, сохраняется и выводится 0
- ▶ если хотя бы одно из значений up , $down$ — ноль, то по переднему фронту clk сохранённое число
 - ▶ увеличивается на 1, если $up = 1$
 - ▶ уменьшается на 1, если $down = 1$
 - ▶ не изменяется, если $up = down = 0$

Пример переиспользования автоматов

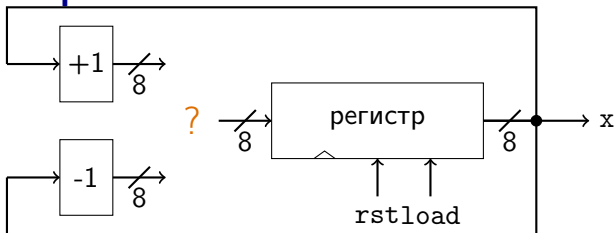


Данные модуля $M_{\uparrow\downarrow}$ очень похожи на данные модулей M_{\uparrow} и M_{\downarrow}

Различия этих трёх модулей состоят только в том, как происходит **управление** изменением этих данных

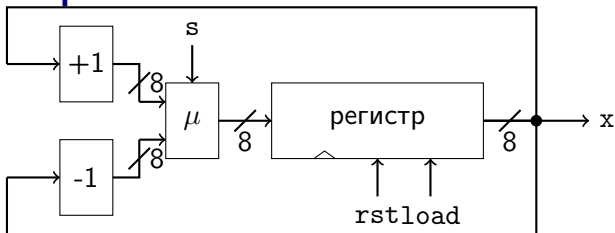
Можно попытаться построить операционный автомат $M_{\uparrow\downarrow}$, совместив правильным образом *почти одинаковые* операционные автоматы модулей M_{\uparrow} и M_{\downarrow}

Пример переиспользования автоматов



При совмещении операционных автоматов “в лоб” некоторые части схемы могут породить точку конфликта: в эту точку, в зависимости от обстоятельств, пересылается одно из значений в нескольких других точках схемы

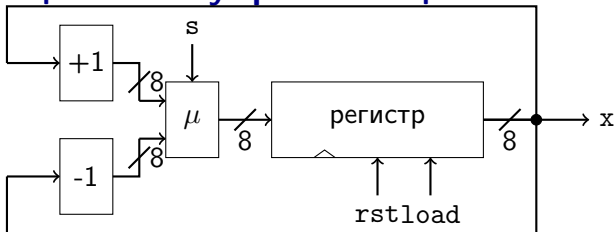
Пример переиспользования автоматов



При совмещении операционных автоматов “в лоб” некоторые части схемы могут породить точку конфликта: в эту точку, в зависимости от обстоятельств, пересылается одно из значений в нескольких других точках схемы

Самый простой способ разрешения такого конфликта — поставить в соответствующую точку **мультиплексор** (μ) и предоставить управляющему автомату решать выставлением соответствующего управляющего сигнала (s), какое именно значение пересылается в точку конфликта

Комбинационный управляющий автомат

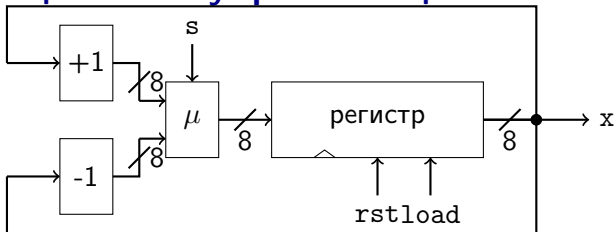


Сигналы `rst` и `load`, как и `s`, отсутствуют в описании всей схемы и являются **управляющими**, а значит, они должны выставляться управляющим автоматом

В общем случае управляющий автомат — довольно нетривиальная подсхема

Однако в некоторых случаях для выставления управляющих сигналов не требуется никаких *состояний* и *переходов*, и управляющий автомат вырождается в **комбинационную** схему

Комбинационный управляющий автомат



Например, для модуля $M_{\uparrow\downarrow}$ с предложенным операционным автоматом подходит такой управляющий автомат (в терминах Verilog):

```
assign s = down;  
assign rst = up && down;  
assign load = up ^ down;
```