|  |  |
| --- | --- |
| ДИСЦИЛИНА | **Схемотехника устройств компьютерных систем Часть 2** |
|  |  |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **вычислительной техники** |
|  |  |
| ВИД УЧЕБНОГО | **Лекция** |
| МАТЕРИАЛА |  |
| ПРЕПОДАВАТЕЛЬ | **Тарасов И.Е.** |
|  |  |
| СЕМЕСТР | 3 |
|  |  |

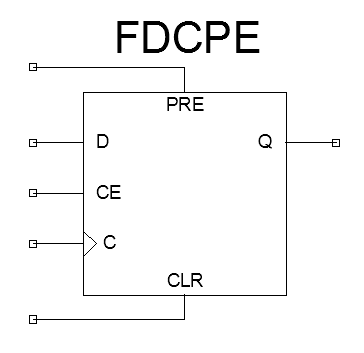
6. Реализация основных синхронных устройств

6.1. Содержание раздела.

Ранее было упомянуто, что современные цифровые схемы должны основываться на синхронных элементах. Это элементы, изменяющие свое состояние не в произвольные моменты времени, а в строго определенный – обычно это фронт тактового сигнала. В данном разделе рассматриваются разновидности синхронных цифровых узлов и способы их проектирования на языках описания аппаратуры.

6.2. Триггер, его разновидности и порядок описания.

Наиболее распространенным синхронным элементом является триггер. Формально, к триггерам относится целый ряд цифровых элементов, которые могут находиться в устойчивом состоянии, однако не все триггеры являются синхронными. На рис. 6.1 показано условное изображение D-триггера, который представляет собой синхронный элемент.



*Рисунок 6.1 Условное изображение D-триггера*

Показанный триггер имеет следующие выводы:

С – вход тактового сигнала;

D – вход данных;

CE – вход разрешения записи (clock enable);

CLR – вход асинхронного сброса (clear);

PRE – вход асинхронной установки в логическую единицу (preset);

R – вход синхронного сброса (reset);

S – вход синхронной установки в логическую единицу (set);

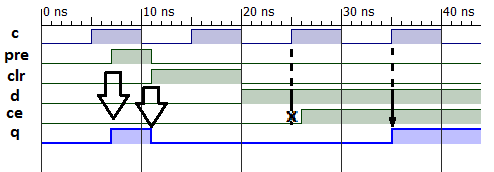
Q – выход данных.

В D-триггере не обязательно использовать все входы. В минимальном варианте достаточно иметь тактовый вход C (также часто обозначается как CLK), и вход данных D. Очевидно, что выход данных Q должен быть всегда.

В целом работу D-триггера можно описать простым правилом:

**«В момент прихода фронта тактового сигнала выход Q принимает состояние входа D».**

Дополнительные входы управления корректируют выполнение этого правила. На рис. 6.2 показаны временные диаграммы работы триггера, у которого присутствуют дополнительные входы управления, в том числе и асинхронные.



*Рисунок 6.2 Временные диаграммы работы триггера*

На диаграмме можно видеть следующие основные события, происходящие с триггером.

1. В момент времени 7 нс появляется сигнал PRE – это сигнал асинхронной установки. Наличие этого сигнала переводит триггер в состояние логической единицы вне зависимости от наличия фронта тактового сигнала.

2. В момент времени 11 нс появляется сигнал CLR — это сигнал асинхронного сброса. Наличие этого сигнала переводит триггер в состояние логического нуля вне зависимости от наличия фронта тактового сигнала.

Если действуют оба сигнала, PRE и CLR, поведение триггера зависит от его реализации. Часто оба асинхронных сигнала не реализуют. Кроме того, асинхронный сброс и установка не рекомендуются к применению, хотя и допустимы с точки зрения цифровой схемотехники.

3. В момент времени 25 нс нет асинхронных сигналов, есть фронт тактового сигнала, но триггер не реагирует на вход данных, поскольку у него неактивен вход CE («разрешение счета»).

4. В момент времени 35 нс триггер находится в наиболее показательном состоянии – нет сигналов PRE, CLR, активен сигнал CE, поэтому происходит запись в триггер того уровня, который присутствует на входе D.

Описание D-триггера на языке Verilog приведено ниже:

module dff( input clk,

input d,

output reg q);

always @ (posedge clk)

q <= d;

endmodule

В VHDL триггер описывается аналогично. Можно обратить внимание на отсутствие спецификатора reg для выходного сигнала, как в Verilog. Синтезатор VHDL определяет необходимость выбора триггера автоматически.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity dff is

Port ( clk : in STD\_LOGIC;

d : in STD\_LOGIC;

q : out STD\_LOGIC);

end dff;

architecture Behavioral of dff is

begin

process(clk)

begin

if rising\_edge(clk) then

q <= d;

end if;

end process;

end Behavioral;

6.3. Асинхронный и синхронный сброс, рекомендуемые практики применения управляющих сигналов.

При рассмотрении работы триггера были показаны варианты сброса и установки. Эти действия имеют приоритет перед записью данных с входа D. При этом сброс и установка могут быть как асинхронными (т.е. происходить сразу при появлении на входе соответствующего сигнала), так и синхронными (т.е. происходить по фронту тактового сигнала). Синтезаторы могут выбирать соответствующий вариант триггера на основе его RTL-представления.

Триггер с синхронным сбросом:

always @(posedge clk)

if (reset)

begin

q <= 0;

end else

begin

q <= d;

end

Триггер с асинхронным сбросом:

always @(posedge clk or posedge reset)

if (reset) begin

q <= 0;

end else if (ce) begin

q <= d;

end

Можно обратить внимание, что для асинхронного сброса в список чувствительности добавлено выражение posedge reset. Это необходимо, чтобы при моделировании такой схемы изменение сигнала reset правильно обрабатывалось программой-симулятором.

Сравнивая синхронный и асинхронный сброс, может показаться, что асинхронный сброс выглядит привлекательнее. Триггер можно сбросить в любой момент, даже коротким импульсом. Для синхронного сброса нужно дождаться фронта тактового сигнала.

Тем не менее, проблема кроется в замечании «даже коротким импульсом». При уменьшении технологических норм это становится проблемой. Логическая модель покажет короткие импульсы, но реальный кристалл из-за вариаций техпроцесса может исказить их длительность и момент прихода на входы триггеров.

Правильно работающая модель с асинхронными сбросами не является доказательством того, что реальный кристалл воспроизведет эти импульсы так же. Некоторые триггеры могут не воспринять короткий сигнал сброса.

Таким образом, рекомендуемый вариант сброса – синхронный.

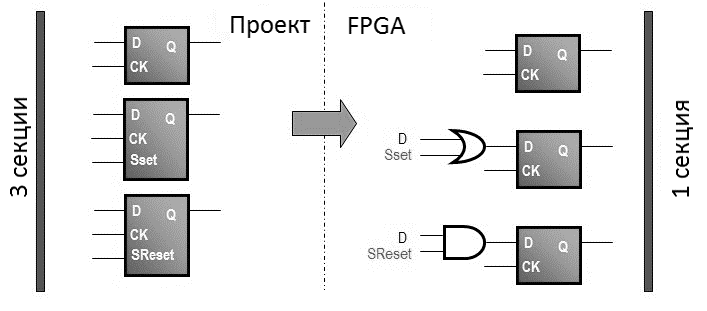
Внешние источники сигналов сброса асинхронны по отношению к проекту. В этом случае требуется *синхронизация* сигналов, что будет рассмотрено в разделе 8.

6.4. Управляющие наборы (control sets) в ПЛИС.

Ячейки современных ПЛИС имеют некоторые особенности организации триггеров. Например, секция в FPGA Xilinx, содержащая 8 триггеров, обобщает все сигналы управления (clk, ce, set/reset). Речь идет не просто о том, что сигнал ce должен иметь высокий активный уровень, а о том, что все триггеры должны использовать один и тот же сигнал. Это сделано производителем для экономии ресурсов, поскольку 8 независимых наборов сигналов займут слишком много места. Нельзя утверждать, что это универсальное правило для всех ПЛИС, однако текущие семейства имеют такую конструкцию ячеек.

Это ограничение не препятствует реализации схем с разными сигналами, просто триггеры могут быть размещены в разных секциях. Кроме этого, синтезатор может использовать приемы преобразования управляющих сигналов в вентили, управляющие данными. Примеры таких автоматических преобразований показаны на рис. 6.3.

Управляющие наборы не являются предметом постоянного внимания при разработке. Тем не менее, наличие ограничений может объяснить, почему схема занимает больше ячеек ПЛИС, чем предполагалось разработчиком.



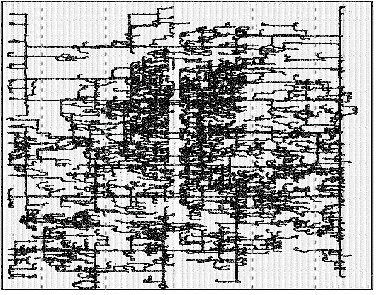
*Рисунок 6.3 Примеры преобразования управляющих сигналов в триггерах ПЛИС*

6.5. Особенности проектирования и моделирования сигнала сброса.

Сброс регистров в заранее определенное состояние является нормальной практикой проектирования цифровых систем. Это устраняет неопределенность, которая могла бы возникнуть при разнообразных процессах в момент включения питания, и широко практикуется при построении самых разных узлов. При разработке СБИС сброс синхронных компонентов является обязательным.

Те же соображения применимы и для FPGA, однако здесь имеется важное дополнение. Сами микросхемы FPGA уже имеют схемы сброса, и установка всех синхронных компонентов в начальное состояние является частью процесса загрузки конфигурации. Поэтому попытка установить все регистры в начальное состояние при старте проекта будет просто дублированием тех действий, которые только что были проведены. Может показаться, что это несущественная деталь, которая просто увеличивает надежность проекта, однако на рис. 6.4 можно ознакомиться с примером трассировки избыточного сигнала сброса. Разумеется, такой сигнал вряд ли сделает трассировку проекта невозможной, однако эти линии могут занять места, которые в ином случае были бы использованы для трассировки более критичных к задержкам сигналов.

Некоторые компоненты ПЛИС не могут быть сброшены. Например, массив памяти не имеет сигнала сброса, который мог бы обнулить все ячейки памяти. Попытка описать сброс для массива памяти вызовет его реализацию на базе триггеров, что в целом нерационально, поскольку блок памяти существенно компактнее по сравнению с набором триггеров такого же логического объема.



*Рисунок 6.4 Примеры трассировки цепи сброса в ПЛИС*

В целом можно придерживаться следующих соображений:

1. Лучшим приемом является отказ от глобального сброса, поскольку он уже реализован в ПЛИС.

2. Локальные сигналы сброса, если они необходимы для проекта, должны быть синхронными и использовать высокий логический уровень.

При поведенческом описании сигналов можно использовать инициализацию. Она будет использована не только для моделирования, но и для загрузки в соответствующий регистр при инициализации ПЛИС.

6.6. Разработка счетчика

Счетчик выполняет последовательное увеличение или уменьшение своего выходного значения по каждому тактовому импульсу. Простейший вариант счетчика показан ниже:

reg [7:0] cnt;

always @ (posedge clk)

cnt <= cnt + 1;

Поскольку для хранения значения счетчика выбрано 8 разрядов, счетчик будет осуществлять циклическое приращение своего значения от 0 до 255, после чего операция 255+1 опять сделает значение cnt равным 0.

В показанном примере счетчик работает строго в диапазоне от 0 до 255. Переключение к новой последовательности происходит естественным образом, в результате переполнения регистра-счетчика.

Если нужно обеспечить счет до определенного значения, которое не может быть обеспечено простым переполнением, следует использовать внутри процесса always условный оператор, загружающий в счетчик 0, если достигнут предел счета, или продолжающий счет в противном случае.

Для счетчиков используются следующие возможности:

– разрешение счета;

– регулируемое направление счета (up/down);

– возможность сброса;

– возможность загрузки.

Счетчик с регулируемым направлением счета приведен в следующем примере:

reg [7:0] cnt;

always @(posedge clk)

if (up\_down)

cnt <= cnt + 1;

else

cnt <= cnt - 1;

Счетчик имеет дополнительный управляющий вход up\_down. Высокий логический уровень на этом входе означает счет на увеличение значения счетчика, а низкий – на уменьшение.

Счетчик с загрузкой имеет дополнительные входы – разрешение загрузки и загружаемые данные. Если на входе разрешения загрузки присутствует активный уровень, то счетчик принимает значение, заданное внешней шиной. Таким образом, с помощью этого интерфейса можно принудительно задать требуемое значение счетчика.

Пример счетчика с синхронным сбросом и загрузкой:

reg [7:0] cnt;

always @(posedge clk)

if (reset)

cnt <= 0;

else if (ce)

if (load)

cnt <= d\_in;

else

cnt <= cnt + 1;

Как и для остальных цифровых модулей на базе ПЛИС, рекомендуется использовать синхронный сброс вместо асинхронного.

Двоичное кодирование является не единственно возможным алгоритмом работы счетчика. Вместо последовательного перебора двоичных значений в процессе работы возможно использование и других кодировок. Например, код Грея (Gray code) имеет то свойство, что для перехода к следующему значению достаточно изменить значение единственного разряда. Это полезно при обработке сигналов, в которых существует вероятность сдвига по времени между отдельными разрядами. Например, при переходе от двоичного состояния 0111 (710) к 1000 (810) из-за неодновременной смены разрядов может появиться состояние 0000, 1111 (или любое другое, в зависимости от порядка смены разрядов). В то же время подобный эффект при использовании кода Грея приведет к максимальной ошибке, равной 1.

Пример реализации счетчика, основанного на коде Грея:

parameter gray\_width = 8;

reg [gray\_width-1:0] binary\_value;

reg [gray\_width-1:0] gray\_value;

always @(posedge clk)

if (reset) begin

binary\_value <= {{gray\_width{1'b0}}, 1'b1};

gray\_value <= {gray\_width{1'b0}};

end

else if (ce) begin

binary\_value <= binary\_value + 1;

gray\_value <=

(binary\_value >> 1) ^ binary\_value;

end

Другой разновидностью кодирования является LFSR (*Linear Feedback Shift Register*) – сдвиговый регистр с линейной обратной связью. Пример 4-разрядного LFSR:

reg [3:0] lfsr;

always @(posedge clk)

if (reset)

lfsr <= 4'h0;

else if (ce) begin

lfsr[3:1] <= lfsr[2:0];

lfsr[0] <= ~^lfsr[4:3];

end

Особенностью кодирования по LFSR является более быстрая смена состояний по сравнению с двоичным счетчиком, поскольку при двоичном кодировании возможна ситуация, когда прибавление единицы сменит все разряды, включая самый старший. На распространение бита переноса по всем разрядам двоичного счетчика тратится дополнительное время, по сравнению со сдвиговыми регистрами, в которых каждый разряд получает свое значение от соседнего разряда. Вдвигаемое в LFSR значение определяется в строке lfsr[0] <= ~^lfsr[4:3], и зависит от разрядности счетчика.

Недостатком счетчика LFSR является меньшее число уникальных состояний по сравнению с двоичным счетчиком той же разрядности.

6.7. Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ, также PWM – Pulse-Width Modulation) является эффективным способом цифрового управления силовыми системами. При этом регулирование мощности осуществляется путем управления отношением времени, в течение которого сигнал включен, ко времени, в течение которого он выключен. Такой способ управления обладает как минимум двумя достоинствами – он удобен для реализации в цифровой системе, и является энергоэффективным, поскольку в ключевом режиме работы управляющий элемент потребляет минимальную мощность (в закрытом состоянии ток через него стремится к нулю, а в открытом стремится к нулю падение напряжения, поскольку сопротивление переключающих элементов стремятся уменьшить).

Модуль ШИМ разрабатывается следующим образом. Входной сигнал d задает число тактов, в течение которых следует удерживать высокий логический уровень на выходе модуля. Внутренний сигнал cnt циклически перебирает состояния от 0 до максимального значения, которое может быть подано в качестве входного. Допустим, что внутренний счетчик является 8-разрядным (т.е. полный цикл счета содержит 256 тактов). Тогда при подаче на вход числа 10 на выходе такого модуля будет логическая единица в течение 10 тактов, а в течение остальных 246 – логический ноль. Увеличивая значение числа, поданного на вход d, можно увеличивать отношение времени включения выходного сигнала к общему времени цикла счета.

module pwm(

input clk,

input [7:0] d,

output pwm

);

reg [7:0] cnt = 0;

always @ (posedge clk)

cnt <= cnt + 1;

assign pwm = (d > cnt) ? 1 : 0;

endmodule

Для демонстрации работы созданного модуля используется следующая тестовая последовательность: формируется тактовый сигнал с периодом 20 нс, и на вход данных подается фиксированное значение 10010. Тестовый модуль pwm\_tb показан ниже:

module pwm\_tb;

// Inputs

reg clk;

reg [7:0] d;

// Outputs

wire pwm;

// Instantiate the Unit Under Test (UUT)

pwm uut (

.clk(clk),

.d(d),

.pwm(pwm)

);

initial begin

// Initialize Inputs

clk = 0;

d = 100;

forever clk = #10 ~clk;

end

endmodule

Для получения периода сигнала ШИМ, не равного целой степени двойки, следует обеспечить изменение счетчика в диапазоне от 0 до необходимого максимального значения.

6.8. Выводы по разделу

Для цифровых схем используется синхронный стиль проектирования:

- один тактовый сигнал для модуля;

- основной триггер – D-триггер (с разрешением счета);

- сброс – синхронный, предпочтительно с активным высоким уровнем.

Счетчик является основой для построения других схем, например, широтно-импульсного модулятора.

Контрольные вопросы:

1. Какие сигналы есть у D-триггера?

2. Какой сброс предпочтительнее для триггера – синхронный или асинхронный? Почему?

3. Сколько разрядов должен иметь счетчик, способный считать до 200?

4. Какое выражение на Verilog необходимо использовать, чтобы временно остановить счетчик?

5. Мощный транзистор необходимо переключать с частотой не менее 100 кГц. Требуется обеспечить погрешность не хуже 0.1%. Какую тактовую частоту необходимо подать на модуль ШИМ, управляющий таким транзистором?