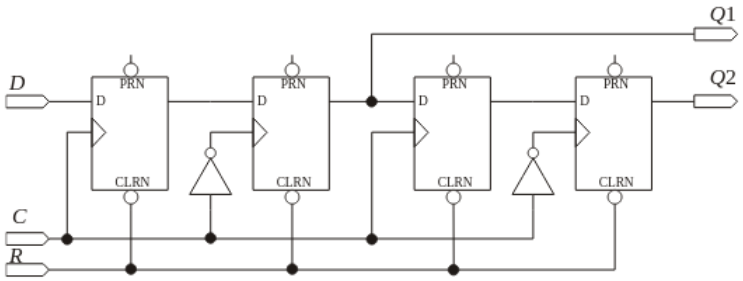
Lab 4

2-разрядный 1-тактовый сдвигающий регистр на D-триггере (DFF)

ДОРИСОВАТЬ ВЫХОДЫ В ПРОМЕЖУТКАХ Q1.1, Q2.2



D-data
R-reset
C-clock signal
Сигнал который синхронизирует работу триггеров. При каждом фронте тактового сигнала данные сдвигаются от одного триггера к следующему.
DFF(Data Flow Flip-Flop) - D-триггер здесь работает с тактовым сигналом (C-clock signal). Он обновляет свое состояние только в момент перехода тактового сигнала. Треугольник включает триггер

Как просимулировать работу регистра:

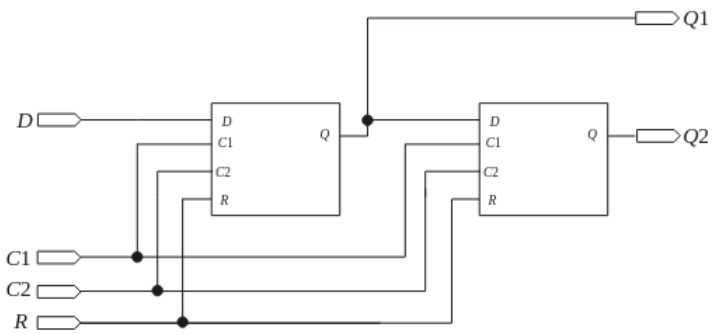
1. **Подготовка к симуляции:**
   - Постройте схему, состоящую из двух D-триггеров, подключённых последовательно: выход первого триггера подается на вход второго триггера.
2. **Сигналы для симуляции:**
   - **Тактовый сигнал (C):** Подайте на вход C периодический сигнал, который будет инициировать сдвиг данных. Важно синхронизировать такты с другими сигналами.
   - **Входные данные (D):** Поочередно подавайте значения (0 или 1) на вход D, чтобы наблюдать за тем, как данные сдвигаются через регистр.
   - **Сброс (R):** Используйте сигнал сброса, чтобы обнулить значения триггеров в начале симуляции или при необходимости.
3. **Шаги симуляции:**
   - В начале симуляции подайте активный сигнал сброса (R = 0), чтобы очистить регистр (Устанавливаем начальное состояние).
   - Установите сигнал сброса в неактивное состояние (R = 1).
   - Начните подавать тактовые импульсы на вход C и изменяйте данные на входе D.
   - Наблюдайте, как данные сдвигаются с входа первого триггера на второй на каждом такте.
4. **Порядок работы регистра:**
   1. **Первый такт:**
      - На вход D1 подается первый бит данных (например, 1).
      - Когда тактовый сигнал C переходит с 0 на 1 (передний фронт), значение на входе D1 записывается в первый триггер (Q1).
      - В результате, Q1 = 1, а Q2 остаётся 0, так как второй триггер ещё не получил данных от первого.
   2. **Второй такт:**
      - На следующий передний фронт тактового сигнала C, значение первого триггера (Q1) сдвигается во второй триггер (Q2).
      - В то же время на вход D1 можно подать новый бит (например, 0).
      - Теперь Q1 = 0 (новое значение на входе D1), а Q2 = 1 (значение, сдвинутое из первого триггера).
   3. **Следующие такты:**

- Процесс продолжается: данные на входе D1 с каждым тактом записываются в первый триггер, а данные из первого триггера сдвигаются во второй триггер. Каждый такт обновляет выходы Q1 и Q2 в зависимости от поступающих данных.

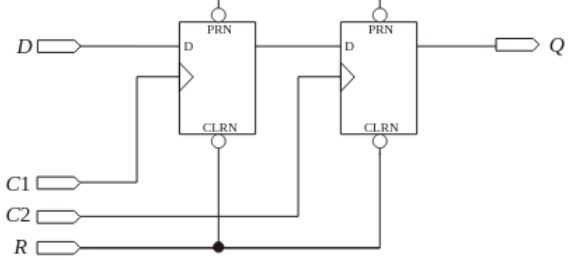
D	C	R	Q1	Q2
1	0	1	0	0
1	1	1	0	0
0	0	1	1	0
0	1	1	1	0
0	0	1	0	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vhd2r1tsdvig IS
    PORT (
        d, c, r : IN STD_LOGIC;
        q2 : INOUT STD_LOGIC;
        q4 : OUT STD_LOGIC);
END vhd2r1tsdvig;
ARCHITECTURE behav OF vhd2r1tsdvig IS

    SIGNAL q1 : STD_LOGIC := '1';
    SIGNAL q3 : STD_LOGIC := '1';
BEGIN
    PROCESS (r, d, c)
    BEGIN
        IF r = '1' THEN
            IF c = '1' THEN
                q1 <= d;
                q3 <= q2;
            ELSIF c = '0' THEN
                q2 <= q1;
                q4 <= q3;
            END IF;
        ELSE
            q1 <= '0';
            q2 <= '0';
            q3 <= '0';
            q4 <= '0';
        END IF;
    END PROCESS;
END behav;
```

2-разрядный 2-тактовый сдвигающий регистр (на D-триггере с двумя тактовыми входами)



C1 & C2-clock



Это двухтактный D-триггер(Дорисовать выход в промежутке)

Описание двухтактного D-триггера

1. **Вход данных (D):**
   - Сигнал данных, который поступает на вход триггера и записывается внутрь триггера в зависимости от тактовых сигналов.
2. **Тактовые сигналы (C1 и C2):**
   - Два тактовых сигнала управляют моментами записи данных в триггер.
   - Тактовый сигнал **C1** обычно управляет захватом данных на входе D.
   - Тактовый сигнал **C2** управляет передачей данных на выход, т.е. в момент его активации данные фиксируются на выходе Q.
3. **Выход (Q):**
   - Сигнал на выходе, который хранит текущее состояние данных после активации C2.
4. **Сброс (Reset, R):**
   - Как и в стандартных триггерах, сброс обнуляет значение триггера, устанавливая его выход в "0".

2-разрядный 2-тактовый сдвигающий регистр — это устройство, которое сдвигает данные на 2 позиции за полный цикл работы. В отличие от обычного регистра, который использует один тактовый сигнал (Clock), здесь используется **два тактовых сигнала** (C1 и C2). Это значит, что данные могут перемещаться не только за один такт, но за два такта, что позволяет более гибко управлять сдвигом информации.

Описание работы:

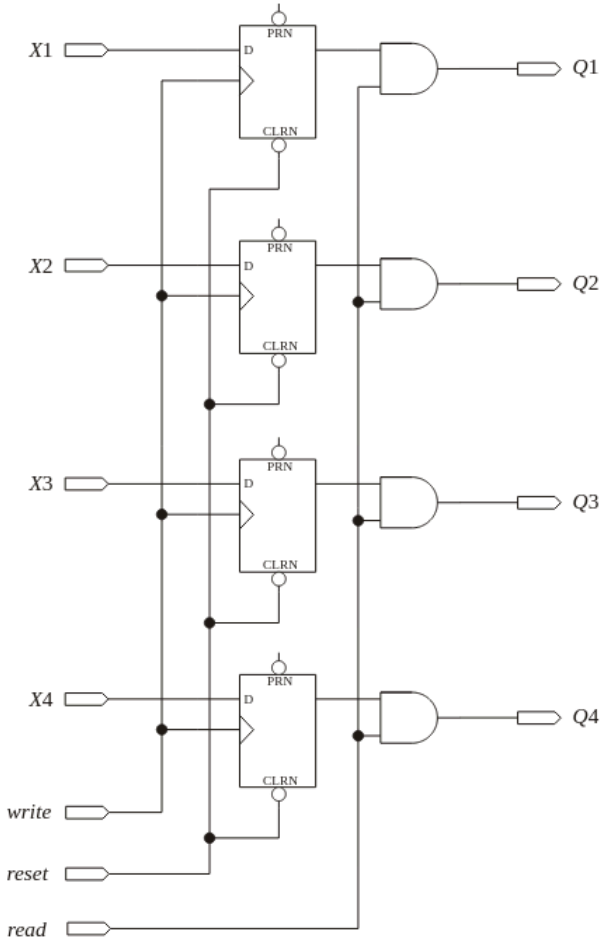
1. **Инициализация (сброс):**
   - В начале работы подаётся сигнал сброса (R = 0), который обнуляет оба триггера (Q1 = 0 и Q2 = 0). После этого сброс деактивируется (R = 1).
2. **Первый такт (C1):**
   - Когда первый тактовый сигнал (C1) переходит с 1 на 0 (передний фронт), и (C2) переходит с 0 на 1 данные с входа D1 записываются в первый триггер (Q1).
3. **Второй такт (C2):**
   - При следующем переходе тактового сигнала C1 и C2, значение первого триггера (Q1) передаётся во второй триггер (Q2).
   - Первый триггер при этом может принять новые данные с входа D1.
   !!C1 и C2 чередуются, если подать 2 одинаковых сигнала ничего не будет!!

D	C1	C2	R	Q1	Q2
1	1	0	1	0	0
1	0	1	1	1	0
0	1	0	1	1	0
0	0	1	1	0	1
0	1	0	1	0	1
0	0	1	1	0	0

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vhd8 IS
    PORT (
        d, c1, c2, r : IN STD_LOGIC;
        q1, q2, q3 : INOUT STD_LOGIC;
        q4 : OUT STD_LOGIC);
END vhd8;

ARCHITECTURE behav OF vhd8 IS
BEGIN
    PROCESS (r, c1, c2, d)
    BEGIN
        IF r = '1' THEN
            IF c1 = '1' THEN
                q1 <= d;
                q3 <= q2;
            END IF;

            IF c2 = '1' THEN
                q2 <= q1;
                q4 <= q3;
            END IF;
        ELSIF r = '0' THEN
            q1 <= '0';
            q2 <= '0';
            q3 <= '0';
            q4 <= '0';
        END IF;
    END PROCESS;
END behav;
```

Простейший регистр памяти

Подать инфу, после включить запись и только после включить чтение



Принцип работы

Сначала, как и везде, чистим, обнуляем триггеры, то есть подаем 0 на reset и ставим в 1.
Работает регистр памяти так, что мы сначала подаем сигналы на входы (X1...X4), ПОСЛЕ ЧЕГО нажимаем запись (переводим write с 0 в 1) и следующим шагом читаем их в выходы (read с 0 в 1) в этот момент выводятся записанные сигналы.

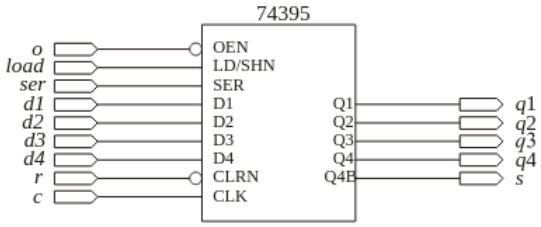
Пример:

X1	X2	X3	X4	write	reset	read	Q1	Q2	Q3	Q4
1	1	0	1	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	0	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY vhd9 IS
    PORT (
        x1, x2, x3, x4, read, write, reset : IN STD_LOGIC;
        q1, q2, q3, q4 : INOUT STD_LOGIC);
END vhd9;

ARCHITECTURE behavior OF vhd9 IS
    SIGNAL i1, i2, i3, i4 : STD_LOGIC;
BEGIN
    PROCESS (x1, write, reset, read)
    BEGIN
        IF reset = '0' THEN
```

```
            i1 <= '0';
        ELSIF reset = '1' THEN
            IF (write = '1') THEN
                i1 <= x1;
            END IF;
        END IF;
    END PROCESS;

    PROCESS (x2, write, reset, read)
    BEGIN
        IF reset = '0' THEN
            i2 <= '0';
        ELSIF reset = '1' THEN
            IF write = '1' THEN
                i2 <= x2;
            END IF;
        END IF;
    END PROCESS;

    PROCESS (x3, write, reset, read)
    BEGIN
        IF reset = '0' THEN
            i3 <= '0';
        ELSIF reset = '1' THEN
            IF write = '1' THEN
                i3 <= x3;
            END IF;
        END IF;
    END PROCESS;

    PROCESS (x4, write, reset, read)
    BEGIN
        IF reset = '0' THEN
            i4 <= '0';
        ELSIF reset = '1' THEN
            IF write = '1' THEN
                i4 <= x4;
            END IF;
        END IF;
    END PROCESS;

    q1 <= i1 AND read;
    q2 <= i2 AND read;
    q3 <= i3 AND read;
    q4 <= i4 AND read;

END behavior;
```

Универсальный регистр(4-разрядный сдвигающий регистр с параллельным входом и параллельным/последовательным выходом)



Как я понял, если мы хотим записать инфу параллельно(из всех d входов), мы включаем `load` '1' и записываем.
Переход `c` из '1' в '0' является началом записи, следующий переход из '0' в '1' концом записи.

`ser` - последовательный ввод информации, информация которую будем записывать при отключенном `load`, выводится она, как я понял, в выход `q1`. Но, проводя симуляцию я запутался, и не понял как она работает.
`r` - reset, очистка выходов.
`o` - не трогаем.

п.10 Синтезируемость и вся дребедень

Ну тут сказать нечего, просто действуем по методичке, тупой копипаст.