Design and Implement 4-bit Universal Shift Register (USR)

Done by:

B.E.Pranav Kumaar
Harsha Dabbara
Dinesh Kumar M.R
Divi Eswar Chowdary

Abstract

- Through this project we want to,
- Understand the concept of universal shift register
- Explore the working of universal shift register
- Demonstrate a coded version of USR in hdl
- Discover the underlying mechanisms of USR through a simulation

Introduction

A Universal shift register is a register which can store or hold data and can perform both the right shift and left shift with parallel load capabilities. Universal shift registers are used as memory elements in computers. A Unidirectional shift register can shift in only one direction. A bidirectional shift register can shift in both the directions. The Universal shift register is a combination design of bidirectional shift register and a unidirectional shift register with parallel load provision.

This logic board has 4 (4 to 1 multiplexer) and 4 D-flipflops. Mux has operations of shift right, left, and parallel load.

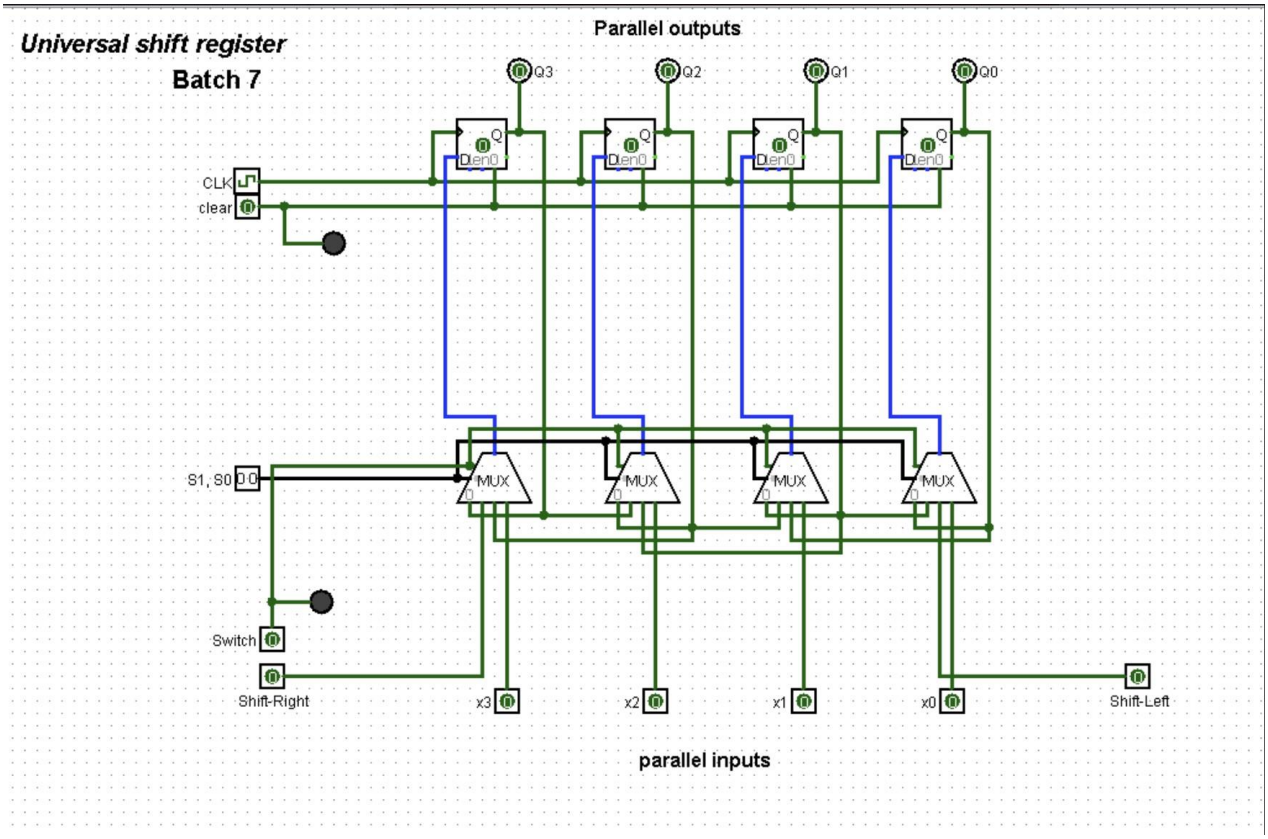
Applications:

- Used in micro-controllers for I/O expansion
- Used as a serial-to-serial converter
- Used as a parallel-to-parallel data converter
- Used as a serial-to-parallel data converter.
- Used in serial – to – serial data transfer
- Used in parallel data transfer.
- Used as a memory element in digital electronics like computers.
- Used in time delay applications
- Used as frequency counters, binary counters, and Digital clocks
- Used in data manipulation applications.

Advantages:

- This register can perform 3 operations such as shift-left, shift-right, and parallel loading.
- Stores the data temporarily with in the register.
- It can perform serial to parallel, parallel to serial, parallel to parallel and serial to serial operations.
- It can perform input-output operations in both the modes serial and parallel.
- A Combination of the unidirectional shift register and bidirectional shift register gives the universe shift register.
- This register acts as an interface between one device to another device to transfer the data.

Logic Circuit



What does 4-Bit Universal Shift Register consist of?

- S0 and S1 are the selected pins that are used to select the mode of operation of this register. It may be shift left operation or shift right operation or parallel mode.

- Pin-0 of first 4×1 Mux is fed to the output pin of the first flip-flop. Observe the connections as shown in the figure.

- Pin-1 of the first 4X1 MUX is connected to serial input for shift right. In this mode, the register shifts the data towards the right.

- Similarly, pin-2 of 4X1 MUX is connected to the serial input for shift-left. In this mode, the universal shift register shifts the data towards the left.

- X0 is the parallel input data given to the pin-3 of the first 4×1 MUX to provide parallel mode operation and stores the data into the register.

- Similarly, remaining individual parallel input data bits are given to the pin-3 of related 4×1 MUX to provide parallel loading.

- Q0, Q1, Q2, and Q3 are the parallel outputs of Flip-flops, which are associated with the 4×1 MUX.

- In parallel data transfer, all the parallel inputs and outputs lines are associated with the parallel load.

- Clear pin clears the register and set to 0.

- CLK pin provides clock pulses to synchronize all the operations.

- In the control state, the information or data in the register would not change even though the clock pulse is applied.

Working

- From the above figure, selected pins the mode of operation of the universal shift register. Serial input shifts the data towards the right and left and stores the data within the register.

- Clear pin and CLK pin are connected to the flip-flop.

- X3, X2, X1, X0 are the parallel inputs Q0, Q1, Q2, and Q3while are the parallel outputs of flip-flops.

- If the selected pins S1= 0 and S0 = 0, then this register doesn’t operate in any mode. That means it will be in a Locked state or no change state even though the clock pulses are applied.

- If the selected pins S1 = 0 and S0 = 1, then this register transfers or shifts the data to left and stores the data.

- If the selected pins S1 = 1 and S0 = 0, then this register shifts the data to right and hence performs the shift-right operation.

- If the selected pins S1 = 1 and S0 = 1, then this register loads the data in parallel. Hence it performs the parallel loading operation and stores the data.

S1	S0	Mode ofOperation
0	0	No Change
0	1	Right Shift
1	0	Left Shift
1	1	Parallel Load

From the above table, we can observe that this register operates in all modes with serial/parallel inputs using 4×1 multiplexers and flip-flops.

Right Shift:

clock	Q3	Q2	Q1	Q0
	0	0	0	0
↑	1	0	0	0
↑	1	1	0	0
↑	1	1	1	0
↑	1	1	1	1

Left Shift:

Clock	Q3	Q2	Q1	Q0
	0	0	0	0
↑	0	0	0	1
↑	0	0	1	1
↑	0	1	1	1
↑	1	1	1	1

Parallel shift

Clock	Q3	Q2	Q1	Q0
	0	0	0	0
↑	0	1	0	1
↑	0	1	0	1

Implementation and Code

The implementation of this 4-Bit Universal Shift Register is done in **HDL**.

The chip is been named as USR (Universal Shift Register) and the respective HDL Code is below:

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;

PARTS:
Twoonemux(i0=false,i1=y1,sel=load,y=in11);

FourOneMux(i0=q31,i1=in11,i2=q21,i3=x3,s0=s0,s1=s1,Y
=in1);

 DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,i3=x2,s0=s0,s1=s1,Y=
in2);

DFF(in=in2,out=q2,out=q21);
```

```
FourOneMux(i0=q11,i1=q21,i2=q01,i3=x1,s0=s0,s1=s1,Y=
in3);


DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41,i3=x0,s0=s0,s1=s1,Y
=in4);


DFF(in=in4,out=q0,out=q01);


Twoonemux(i0=false,i1=y2,sel=load,y=in41);
}
```

Here, DFF is a built-in chip that is present in the nand2tetris folder(tools/builtIn/DFF.hdl).

The code for **TwooneMux:**

```
CHIP Twoonemux {
    IN i0, i1, sel;
    OUT y;
```

```
    PARTS:
      Not(in=sel,out=nsel);
      And(a=i0,b=nsel,out=w1);
      And(a=i1,b=sel,out=w2);
      Or(a=w1,b=w2,out=y);
}
```

The code for **FourOneMux:**

```
CHIP FourOneMux {
    IN i0,i1,i2,i3,s0,s1;
    OUT Y;



   PARTS:
       Twoonemux(i0=i0,i1=i1,sel=s0,y=w1);
        Twoonemux(i0=i2,i1=i3,sel=s0,y=w2);
        Twoonemux(i0=w1,i1=w2,sel=s1,y=Y);
}
```

For Shift-Left Operation:

The condition is s0=0 and s1=1;

Input pins

Name	Value
s0	0
s1	1
x3	0
x2	0
x1	0
x0	0
load	1
y1	0
y2	1

Output pins

Name	Value
q1	0
q2	0
q3	0
q0	1

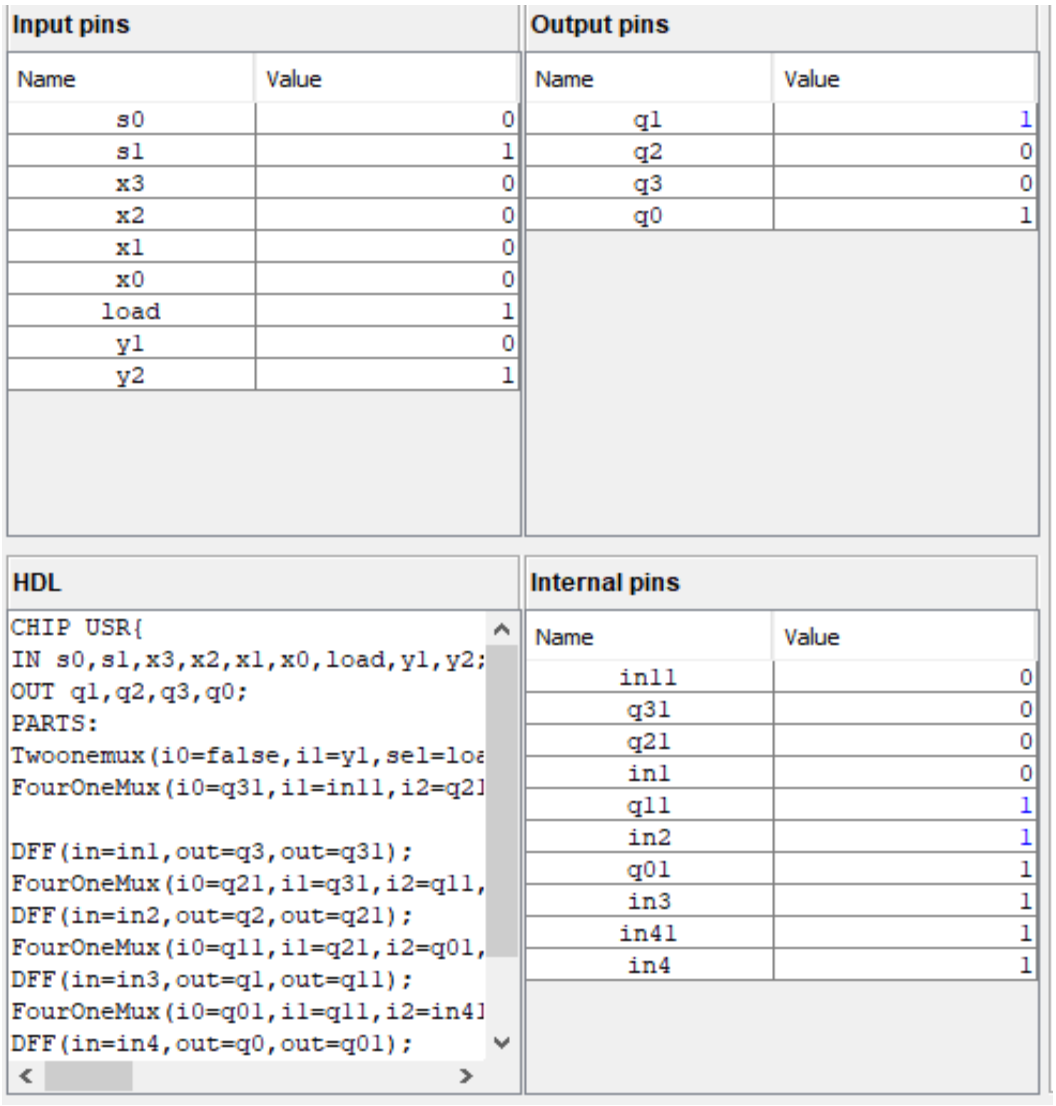
HDL

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;
PARTS:
Twoonemux(i0=false,i1=y1,sel=loa
FourOneMux(i0=q31,i1=in11,i2=q21

DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,
DFF(in=in2,out=q2,out=q21);
FourOneMux(i0=q11,i1=q21,i2=q01,
DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41
DFF(in=in4,out=q0,out=q01);
```

Internal pins

Name	Value
in11	0
q31	0
q21	0
in1	0
q11	0
in2	0
q01	1
in3	1
in41	1
in4	1



Input pins

Name	Value
s0	0
s1	1
x3	0
x2	0
x1	0
x0	0
load	1
y1	0
y2	1

Output pins

Name	Value
q1	1
q2	0
q3	0
q0	1

HDL

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;
PARTS:
Twoonemux(i0=false,i1=y1,sel=loa
FourOneMux(i0=q31,i1=in11,i2=q21

DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,
DFF(in=in2,out=q2,out=q21);
FourOneMux(i0=q11,i1=q21,i2=q01,
DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41
DFF(in=in4,out=q0,out=q01);
```

Internal pins

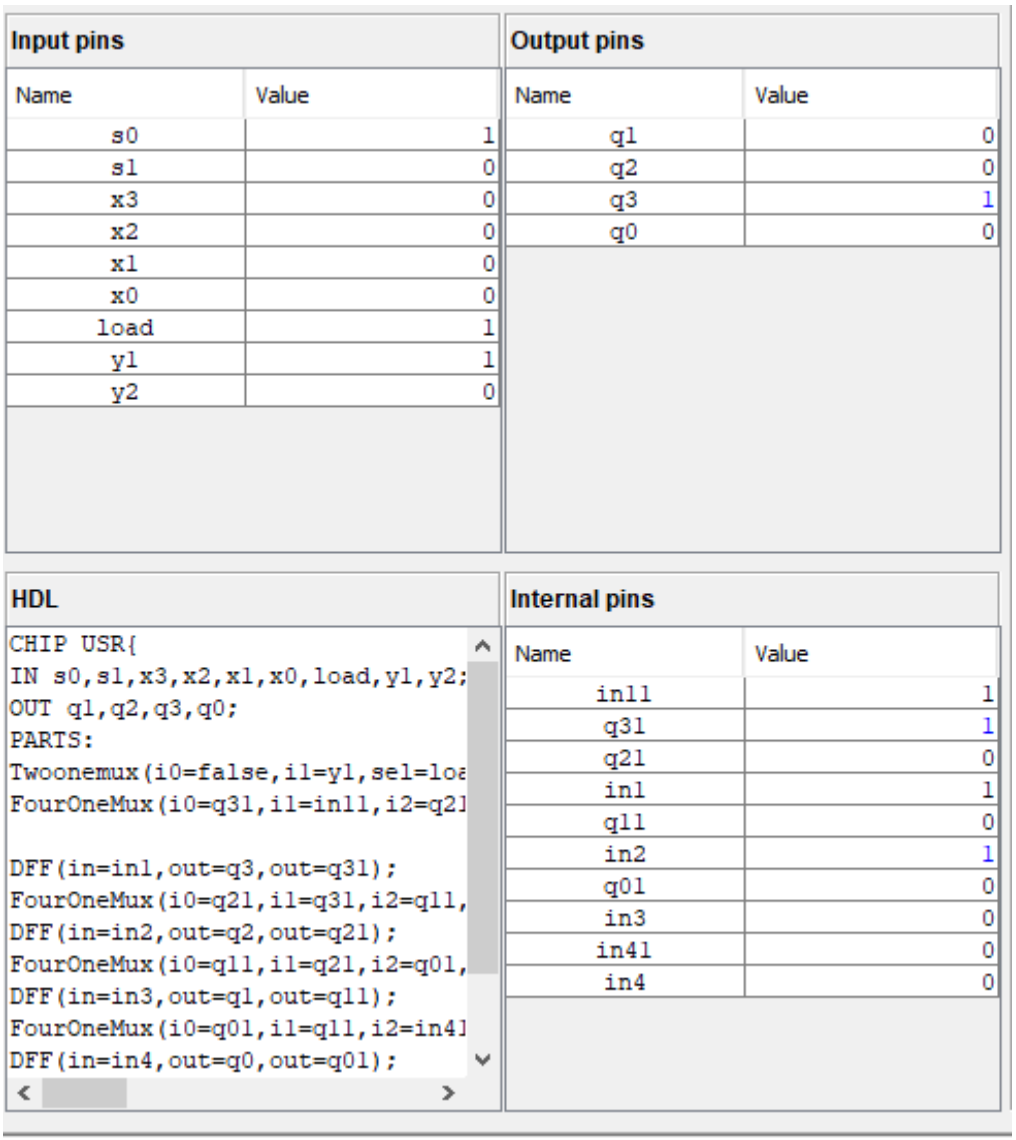
Name	Value
in11	0
q31	0
q21	0
in1	0
q11	1
in2	1
q01	1
in3	1
in41	1
in4	1

From the above two images we can clearly say that the value is been shifted from **q0 to q1 i.e.; it's shifting towards left.**

Similarly, we can do this for other two outputs **(q2 and q3)**.

For Shift-Right Operation:

The condition is s0=1 and s1=0;



Input pins

Name	Value
s0	1
s1	0
x3	0
x2	0
x1	0
x0	0
load	1
y1	1
y2	0

Output pins

Name	Value
q1	0
q2	0
q3	1
q0	0

HDL

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;
PARTS:
Twoonemux(i0=false,i1=y1,sel=loa
FourOneMux(i0=q31,i1=in11,i2=q21

DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,
DFF(in=in2,out=q2,out=q21);
FourOneMux(i0=q11,i1=q21,i2=q01,
DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41
DFF(in=in4,out=q0,out=q01);
```

Internal pins

Name	Value
in11	1
q31	1
q21	0
in1	1
q11	0
in2	1
q01	0
in3	0
in41	0
in4	0

Input pins

Name	Value
s0	1
s1	0
x3	0
x2	0
x1	0
x0	0
load	1
y1	1
y2	0

Output pins

Name	Value
q1	0
q2	1
q3	1
q0	0

HDL

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;
PARTS:
Twoonemux(i0=false,i1=y1,sel=loa
FourOneMux(i0=q31,i1=in11,i2=q21

DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,
DFF(in=in2,out=q2,out=q21);
FourOneMux(i0=q11,i1=q21,i2=q01,
DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41
DFF(in=in4,out=q0,out=q01);
```

Internal pins

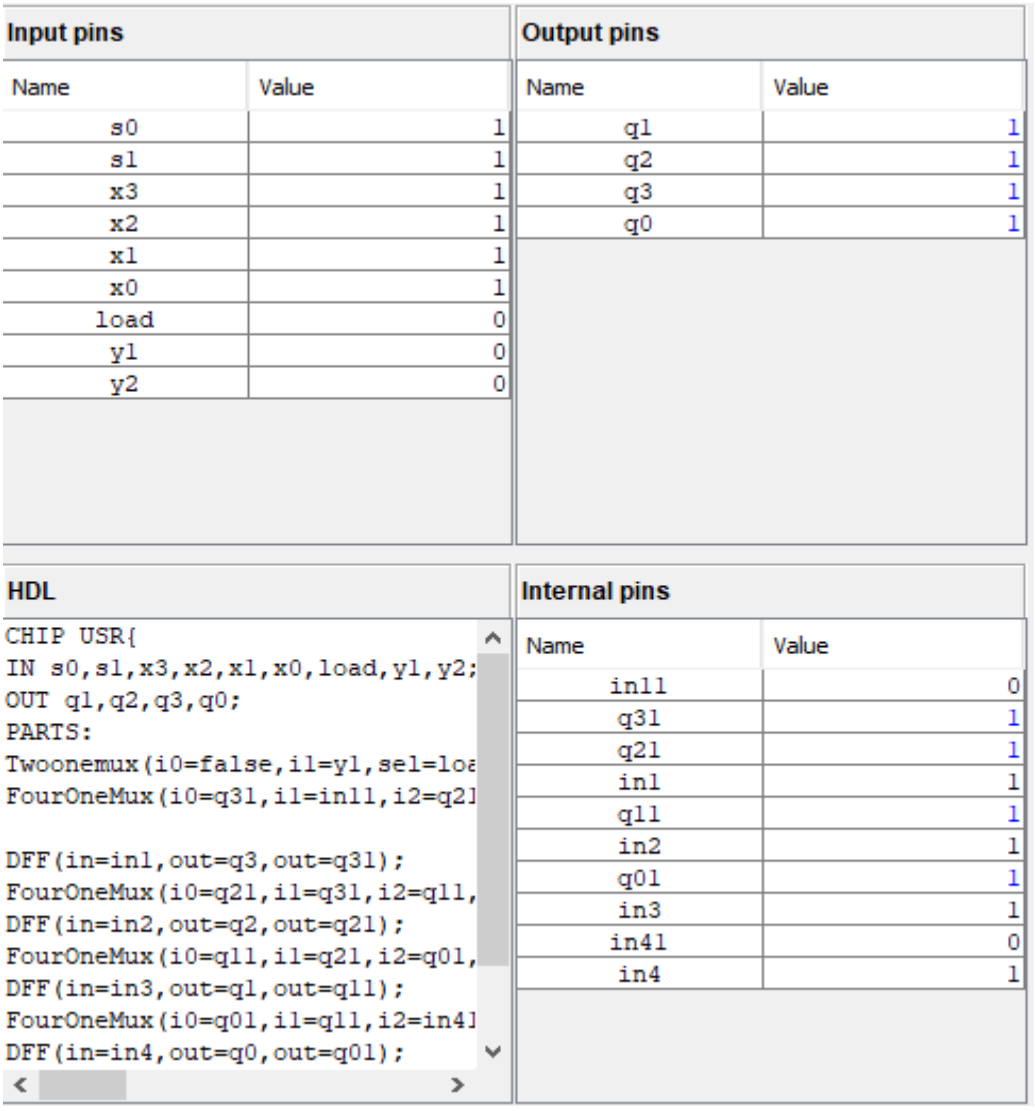
Name	Value
in11	1
q31	1
q21	1
in1	1
q11	0
in2	1
q01	0
in3	1
in41	0
in4	0

From the above two images we can clearly say that the value is been shifted from **q3 to q2 i.e.; it's shifting towards right.**

Similarly, we can do this for other two outputs **(q1 and q0)**.

For Parallel Load Operation:

The condition is s0=1 and s1=1;



Input pins

Name	Value
s0	1
s1	1
x3	1
x2	1
x1	1
x0	1
load	0
y1	0
y2	0

Output pins

Name	Value
q1	1
q2	1
q3	1
q0	1

HDL

```
CHIP USR{
IN s0,s1,x3,x2,x1,x0,load,y1,y2;
OUT q1,q2,q3,q0;
PARTS:
Twoonemux(i0=false,i1=y1,sel=loa
FourOneMux(i0=q31,i1=in11,i2=q21

DFF(in=in1,out=q3,out=q31);
FourOneMux(i0=q21,i1=q31,i2=q11,
DFF(in=in2,out=q2,out=q21);
FourOneMux(i0=q11,i1=q21,i2=q01,
DFF(in=in3,out=q1,out=q11);
FourOneMux(i0=q01,i1=q11,i2=in41
DFF(in=in4,out=q0,out=q01);
```

Internal pins

Name	Value
in11	0
q31	1
q21	1
in1	1
q11	1
in2	1
q01	1
in3	1
in41	0
in4	1

From the outputs we can clearly say that the USR is in loaded state.

Summary

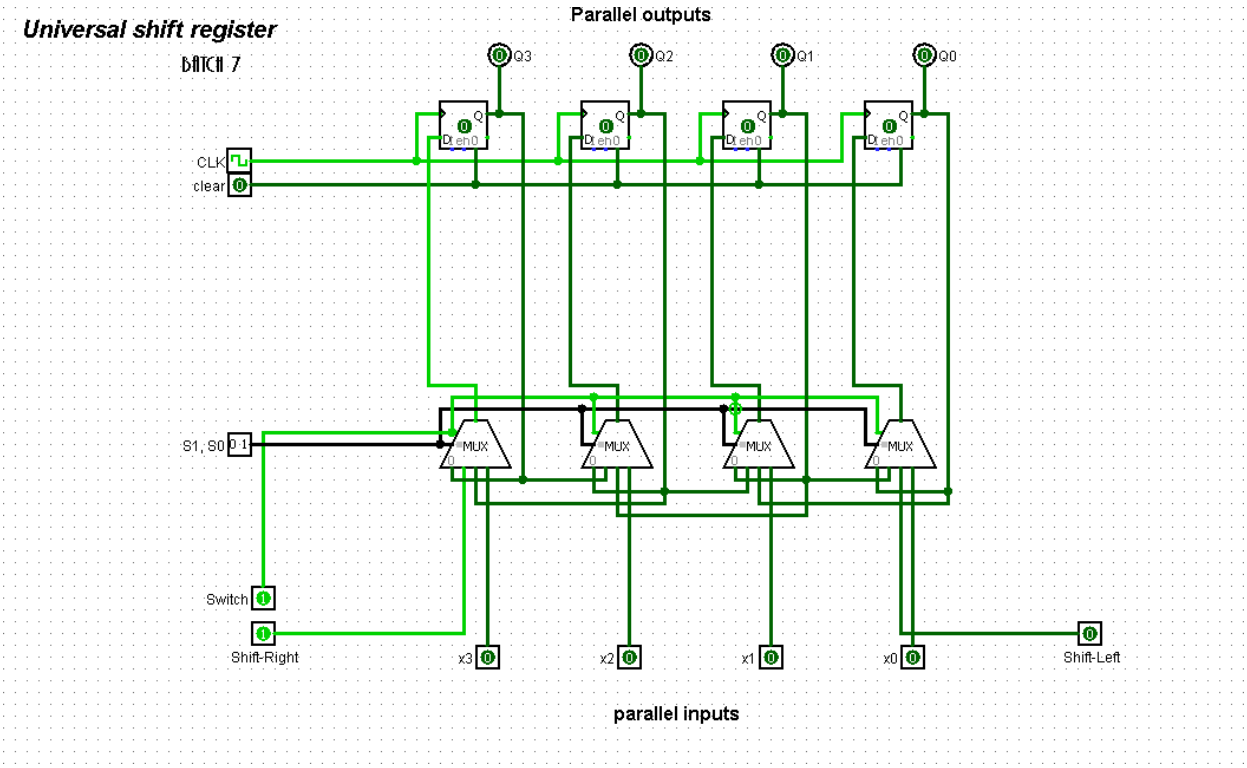
In digital electronics, shift registers are the sequential logic circuits that can store the data temporarily and provides the data transfer towards its output device for every clock pulse. These are capable of transferring/shifting the data either towards the right or left in serial and parallel modes. Based on the mode of input/output operations, shift registers can be used as a serial-in-parallel-out shift register, serial-in-serial-out shift register, parallel-in-parallel-out shift register, parallel-in-parallel-out shift register. Based on shifting the data, there are universal shift registers and bidirectional shift registers.

A register that can store the data and /shifts the data towards the right and left along with the parallel load capability is known as a universal shift register. It can be used to perform input/output operations in both serial and parallel modes. Unidirectional shift registers and bidirectional shift registers are combined together to get the design of the universal shift register. It is also known as a parallel-in-parallel-out shift register or shift register with the parallel load.

Universal shift registers are capable of performing 3 operations as listed below:

- Parallel load operation – stores the data in parallel as well as the data in parallel
- Shift left operation – stores the data and transfers the data shifting towards left in the serial path
- Shift right operation – stores the data and transfers the data by shifting towards right in the serial path.

Hence, Universal shift registers can perform input/output operations with both serial and parallel loads.



References

Definition: https://www.geeksforgeeks.org/www.electronics-tutorials.ws/

Circuit Diagram: https://www.google.com/

Simulation: Logisim Application

Nand2Tetris

Referred Articles

https://www.sciencedirect.com/