**DATAPATH COMPONENT**

Datapath component (RTL component : Register Transfer Level) is a circuit which stores/process data.

Datapath: Circuits has datapath components.

Register is a datapath component.

Diagram

Description automatically generated with medium confidence

Register with Parallel Load

Add 2x1 MUX to front of each flip-flop

Register’s load input selects MUX input to pass

* load = 0 🡪 existing flip-flip value
* load = 1 🡪 new input value

Diagram

Description automatically generated



A picture containing text, antenna

Description automatically generated

When load = 0, I am storing the data.  
When load = 1, I am taking new data from input lines.

Remember: at each clock cycle, we have to take an action

Register Example Using the Load Input: Weight Sampler

Diagram

Description automatically generated

Diagram, schematic

Description automatically generated Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generated Diagram, schematic

Description automatically generated

Buses

Diagram, schematic

Description automatically generatedN-bit bus: N wires to carry N-bit data item

* Circuit drawings can become cluttered

Convention for drawing buses:

* Single bold line and/or small angled line across

A picture containing text, antenna

Description automatically generated

Diagram

Description automatically generatedRegister Example: Above-Mirror Display

T : temperature  
A : average fuel consumption  
I : instantaneous fuel consumption  
M : miles remaining

C : data that is one of these 4

We should activate the correct register for the type of the C.

Diagram

Description automatically generated

You can set x and y to 01 to see 0001010 at the output D.

Register Example: Computerized Checkboard

Graphical user interface, text

Description automatically generated

We use decoder to activate the proper register. To indicate which register stores the data.

A picture containing diagram

Description automatically generated

Diagram

Description automatically generated with medium confidence

Data is going to be feed to all registers but only one of them will be active.

**Shift Register (Kayıkçı)**

1101 -------shift right-------> 0110 (rightmost bit is dropped, assume 0 shifted into leftmost bit)

0100 🡪 0010 🡪 0001 🡪 0000

4 bit right shift register:

Diagram

Description automatically generated

shr : shift right  
shr\_in : shift right input

A picture containing text, shoji

Description automatically generated

Diagram

Description automatically generated



e, s1, s0, c come from car computer. User controls which data should be shown in the mirror 🡪 x, y

I am sending 1 bit for each clock signal.

If we set s1s0 to 01, A will be active. We are saying that “For 8 different bits (cycles), I will set data to A.”. After 8 clock cycle, my data is gonna be stored in register 1 (A).

For example our data is 00010000 for average consumption (A). We will set s1s0 to 01 bc we want to activate register 1.

At first clock signal reg1’s output will be:

* 0 \_ \_ \_ \_ \_ \_ \_

At second clock signal:

* 0 0 \_ \_ \_ \_ \_ \_

At the end:

* 0001 0000

8 bit boyunca s1s0, 01 olarak kalacak. Eğer 8 bit sonunda bir şey gönderilmiyorsa e (enable) 0 yapılır. If e is 0, all of the outputs of decoder are 0. If e = 0, registers are not activated. Registers store the same data.

At 8 clock cycles, I will shift the data step by step because data cable (black c) is 1 wire cable. There is no parallel data. So there is going to be 1 data at each time in the cable.

**Multitask Register**

Tasks:

* Keep the current state (maintain)
* Parallel load
* Shift right
* Shift left

We use 4x1 MUX:

|  |  |  |
| --- | --- | --- |
| **s1** | **s0** | **operation** |
| 0 | 0 | maintain |
| 0 | 1 | load |
| 1 | 0 | shift right |
| 1 | 1 | shift left |

Diagram

Description automatically generated

Üstteki s3, s2, s1, s0 🡪 I3 , I2, I1, I0 olacak.

Diagram

Description automatically generated

We can have separate input for each task if we want:

* ld : load
* shr : shift right
* shl : shift left

If none of them is chosen, maintain state will be active.

A picture containing diagram

Description automatically generated



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ld | shr | shl | operation | s1 | s0 |
| 0 | 0 | 0 | maintain | 0 | 0 |
| 0 | 0 | 1 | shift left | 1 | 1 |
| 0 | 1 | 0 | shift right | 1 | 0 |
| 0 | 1 | 1 | shift right | 1 | 0 |
| 1 | 0 | 0 | load | 0 | 1 |
| 1 | 0 | 1 | load | 0 | 1 |
| 1 | 1 | 0 | load | 0 | 1 |
| 1 | 1 | 1 | load | 0 | 1 |

shift right has priority over shift left.

load has priority over shit right and shift left.

s0 : ld + ld’shr’shl

s1 : ld’shr + ld’shr’shl = ld’(shr + shr’shl) = ld’(shr + shl) = ld’shr + ld’shl

**Design Registers**

* STEP 1: Define the MUX size
* STEP 2: Create MUX operation table
* STEP 3: Connect MUX inputs
* STEP 4: Map control lines

EXAMPLE: Create a register (4 bits) for load, shift left, clear, and set operations using the control inputs (ld, shl, clr, set):

clr will load 0  
set will load 1

We have 5 operations, 5th one is maintain. Maintain is default.

Define the MUX size: We need 8x1 MUX because we have 5 tasks.

Create MUX table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s2 | s1 | s0 | operation | inputs of the MUX |
| 0 | 0 | 0 | maintain | 0 |
| 0 | 0 | 1 | load | 1 |
| 0 | 1 | 0 | shift left | 2 |
| 0 | 1 | 1 | clear | 3 |
| 1 | 0 | 0 | set | 4 |
| 1 | 0 | 1 | maintain | 5 |
| 1 | 1 | 0 | maintain | 6 |
| 1 | 1 | 1 | maintain | 7 |

Connect MUX inputs

Diagram

Description automatically generated

Q-1 🡪 Sonraki registerın outputu, sonraki register yoksa 🡪 shl\_in

Map control lines

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ld | shl | set | clr | s2 | s1 | s0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | … | … | … |
| 0 | 1 | 0 | 1 |  |  |  |
| 0 | 1 | 1 | 0 |  |  |  |
| 0 | 1 | 1 | 1 |  |  |  |
| 1 | 0 | 0 | 0 |  |  |  |
| 1 | 0 | 0 | 1 |  |  |  |
| 1 | 0 | 1 | 0 |  |  |  |
| 1 | 0 | 1 | 1 |  |  |  |
| 1 | 1 | 0 | 0 |  |  |  |
| 1 | 1 | 0 | 1 |  |  |  |
| 1 | 1 | 1 | 0 |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |

You have to define priority for some cases. At the end you get equation for each s2, s1, and s0.

We have created a datapath component.

OTHER DATAPATH COMPONENTS

* Adders
* Subtractor
* Comparator
* Incrementer
* ALU

Son slayt, sayfa 21’den başlıyor.