S 150

omework # 9.

oblem 1: Stream Processor.

) Identify all missing control signals. Make a list of them. Indicate what they do.

ist of needed control signals:

SourceNext: [Output of the controller. Signal to request the source for next data value.

SourceReady: [Input to controller. Tells the controller that the source has new data to be loaded.

DestinationReady: [Input to the controller. Tells the controller that the destination is ready to receive the next data value.

DestinationNext: [Output of the controller. Tells the destinati to load the new output value.

ShiftLeftOutputEnable: [Output of the controller. Enables the shiftLeft module to drive bus C.

ShiftLeftStart: [Output of the controller. Makes the shiftLeft module start the operation.

- ShiftLeftOp : Output of the controller. Tells the ShiftLeft module how many times to shift.
- ShiftLeftDone : Input to the controller. Signals the controller that the ShiftLeft module finished the operation.

- ShiftRightOutputEnable : Output of the controller. Makes the ShiftRight module drive the C bus.
- ShiftRightStart : Output of the controller. Makes the shift Right module start the operation.
- ShiftRightOp : Output of the controller. Tells the shift Right module how many times to shift.
- ShiftRightDone : Input to the controller. Signals the controller that the shift Right module finished its operation.

- InputRegEnable : Output of the controller. Loads new value from the source.
- InputRegOE_A
- InputRegOE_B : Outputs of the controller. Make the input register drive bus A or B respectively.

- OutputRegEnable : Output of the controller. Loads register with value of bus C.

- R_OE_A,
- S_OE_A,
- T_OE_A,
- U_OE_A : Output of the controller. Make register R, S, T or U drive bus A.

- R_OE_B,
- S_OE_B,
- T_OE_B,
- U_OE_B

Output of the controller. Make register R, S, T or U drive bus B.

- R_Input Enable,
- S_Input Enable,
- T_Input Enable,
- U_Input Enable.

Output of the controller. Enable register R, S, T or U to load a new value.

- R_Input Mux select,
- S_Input Mux select,
- T_Input Mux select,
- U_Input Mux select

Output of the controller. Select which bus has access to the input of registers R, S, T and U.

- Add or sub : Output of the controller. Determines the operation of the Arithmetic Module
- Arithmetic OE: Output of the controller. Makes the Arithmetic module drive bus C.

- A equal B : Input to the controller. Tells the controller whether A == B or not.
- A greater B : Input to the controller. Tells the controller whether A > B or not.

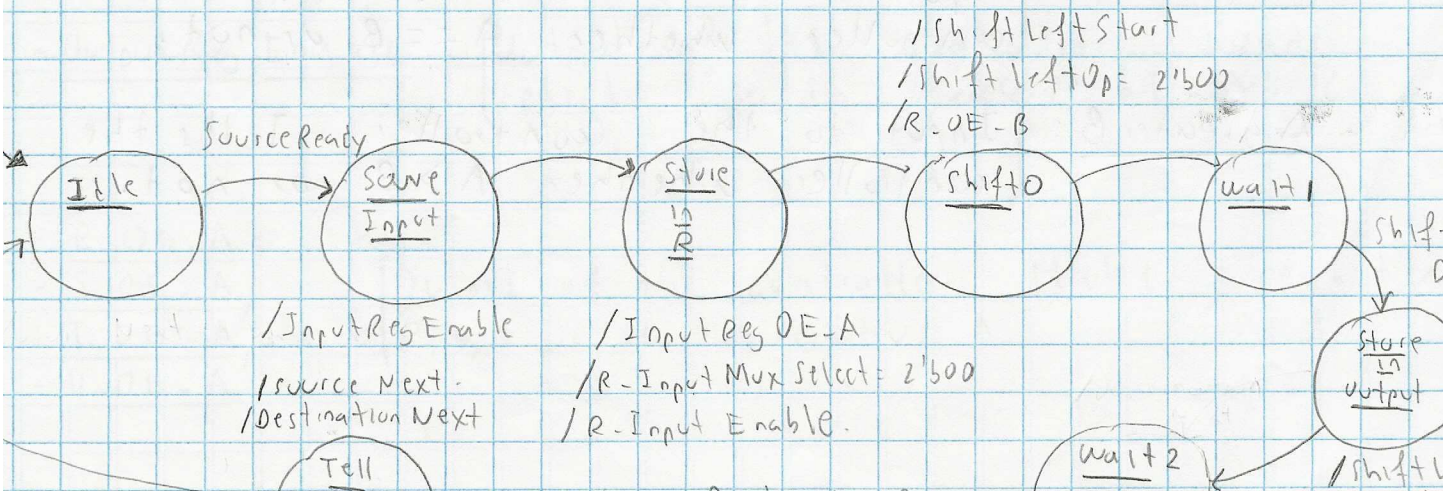
b) Give the RTL to transfer the stream from the input to the output.

```
if (SourceReady == 1)        //New data is available
{
    Input  ← Source ;        InputRegEnable = 1'b1

    R      ← Input ;         InputRegOE_A = 1'b1
                             R_InputMuxSelect = 2'b00  "
                             R_InputEnable = 1'b1.

    Output ← ShiftLeft(R,0)  R_OE_B = 1'b1
                             ShiftLeftOp = 2'b00   "0 sh
                             ShiftLeftStart = 1'b1.  "wa
                             ShiftLeftOutputEnable = 1'b
                             OutputRegEnable = 1'b
}
```

Controller STG



Specify the RTL, Control points and STG for a streaming operation with clipping. R has a maximum value.

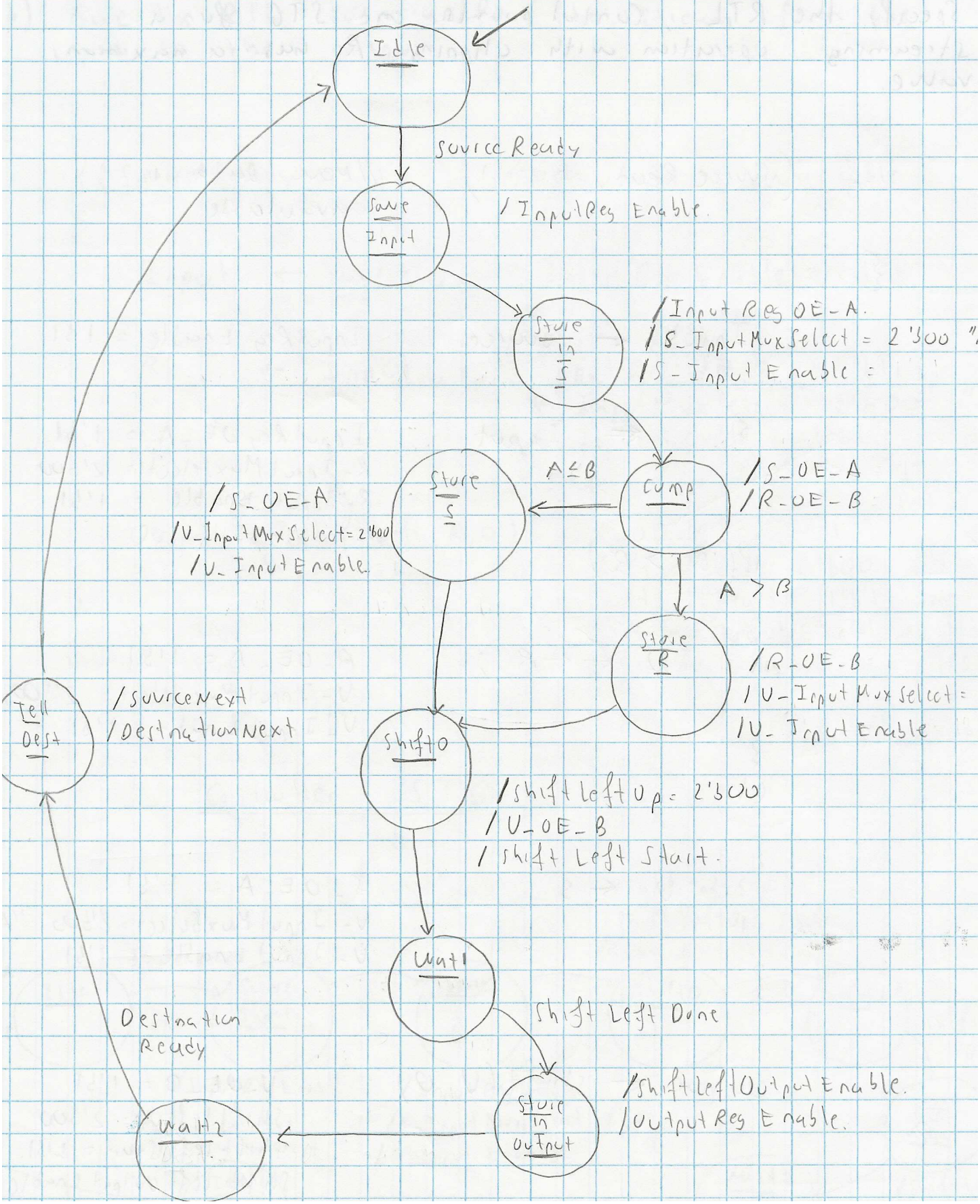
```
If (Source Ready == 1)        // New Data is available
{
    Input ← Source.           InputReg Enable = 1'b1

    S     ← Input             InputReg OE_A = 1'b1.
                              R-Input Mux select = 2'b00
                              R-Input Enable = 1'b1

    if( S > R )
    {
        U ← R ;               R_OE_A = 1'b1.
                              U-Input Mux select = 2'b00
                              U_Input Enable = 1'b1
    }
    else
    {
        U ← S ;               S_OE_A = 1'b1
                              U-Input Mux Select = 2'b00
                              U-Input Enable = 1'b1
    }

    output ← shift(U, 0) ;    U_OE_B = 1'b1.
                              shift left Op = 2'b00
                              shift left start = 1'b1.
                              shift left output Enable
}
```

Controller STG.



d) Specify the RTL, Control Points and STG for a stream operation that averages the new input with the three previous inputs.

Assuming the previous values are in STU

With the oldest in U.

```
if( SourceReady == 1)          // Source has new value
{
    input ← source ;           InputRegEnable = 1'b1;

    R     ← input + U ;        R_InputMuxSelect = 2'b10  "C"
                               R_InputEnable
                               InputRegOE_A = 1'b1
                               U_OE_B = 1'b1
                               AddorSub = 1'b0   "Add"
                               ArithmeticOE = 1'b1

    R ← R + T ;                R_InputMuxSelect = 2'b10  "C"
                               R_InputEnable = 1'b1.
                               R_OE_A = 1'b1
                               T_OE_B = 1'b1
                               AddorSub = 1'b0   "Add"
                               ArithmeticOE = 1'b1

    R ← R + S ;                R_InputMuxSelect = 2'b10  "C
                               R_InputEnable = 1'b1
                               R_OE_A = 1'b1.
                               S_OE_B = 1'b1
                               AddorSub = 1'b0   "Add"
                               ArithmeticOE = 1'b1
```

```
U ← T;
T ← S;
```

U_Input Mux Select = 2'b00 "A"
T_Input Mux Select = 2'b01 "B"
U_Input Enable = 1'b1
T_Input Enable = 1'b1
S_OE_B = 1'b1
T_OE_A = 1'b1

```
S ← Input;
```

S_Input Mux Select = 2'b00 "A"
S_Input Enable = 1'b1
InputReg OE_A = 1'b1

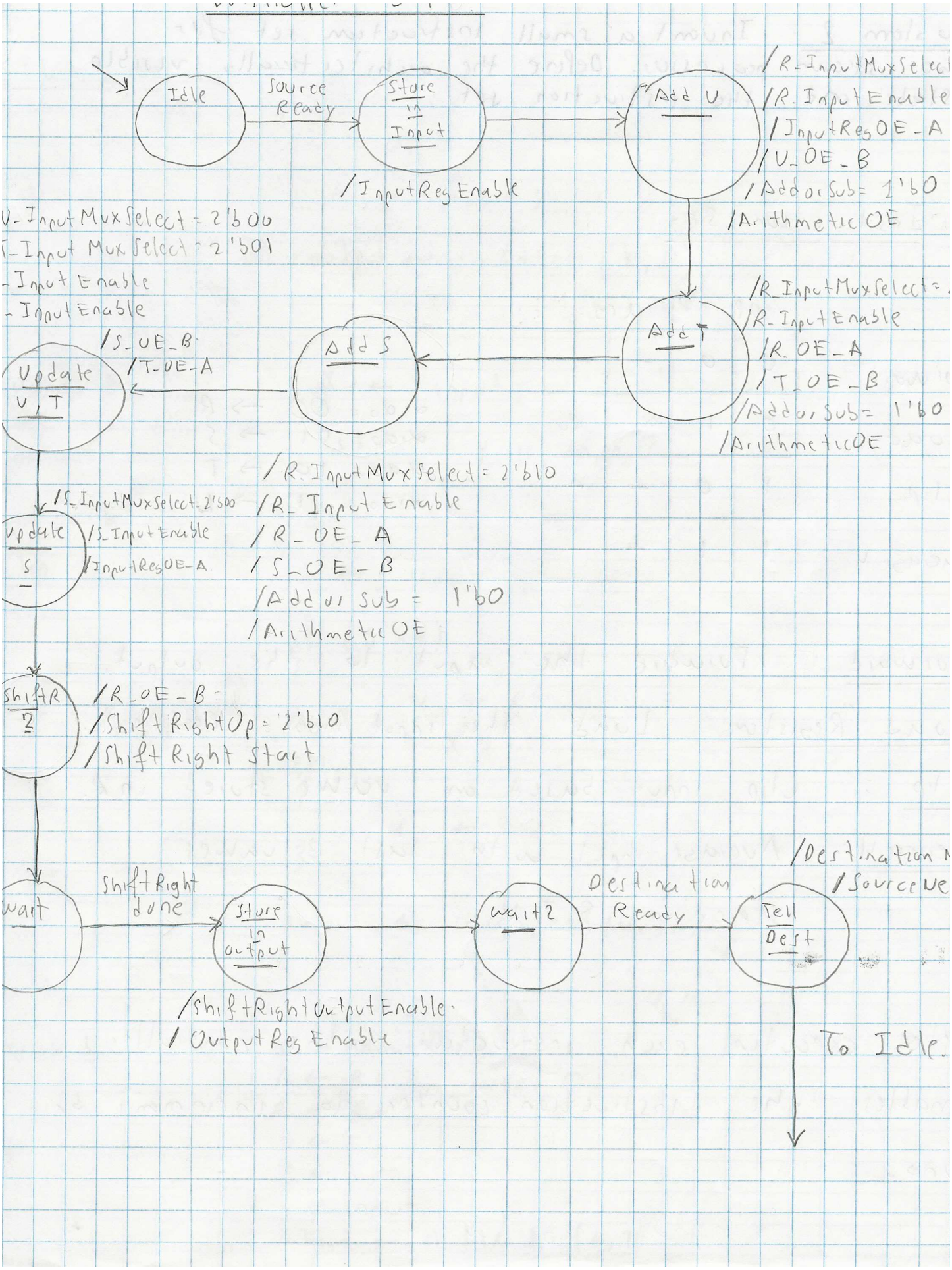
```
// At this point R contains the sum of
   the new value and the previous
   3.
// Also, S, T and U contain now the
   new previous 3 values.

Output ← shiftRight(R,2);
```

R_OE_B = 1;
ShiftRightOp = 2'b10
shiftRight Start = 1
ShiftRight OutputEnable =
OutputReg Enable = 1

```
}
```

The State Transition Graph of the controller that implements this operation is shown in the next page:


Idle
Source Ready
Store in Input
/Input Reg Enable
Add U
/R_Input Mux Select
/R_Input Enable
/Input Reg OE_A
/U_OE_B
/Add or Sub = 1'b0
/Arithmetic OE
U_Input Mux Select = 2'b00
T_Input Mux Select = 2'b01
_Input Enable
_Input Enable
Add T
/R_Input Mux Select =
/R_Input Enable
/R_OE_A
/T_OE_B
/Add or Sub = 1'b0
/Arithmetic OE
Add S
/S_OE_B
/T_OE_A
Update U, T
/R_Input Mux Select = 2'b10
/R_Input Enable
/R_OE_A
/S_OE_B
/Add or Sub = 1'b0
/Arithmetic OE
/S_Input Mux Select = 2'b00
/S_Input Enable
/Input Reg OE_A
Update S
ShiftR
/R_OE_B
/Shift Right Op = 2'b10
/Shift Right Start
Wait
Shift Right done
Store in output
/Shift Right Output Enable
/Output Reg Enable
Wait2
Destination Ready
Tell Dest
/Destination
/Source
To Idle

oblem 2: Invent a small instruction set for he stream processor. Define the architecturally visible state and the instruction set.

Instruction Set

	OP		arg	
rward	0	0	—	—
oad	0	1	α_1	α_0
lip	1	0	—	—
erage 4	1	1	—	—

$\alpha_1\alpha_0 = 00 \rightarrow R$
$\alpha_1\alpha_0 = 01 \rightarrow S$
$\alpha_1\alpha_0 = 10 \rightarrow T$
$\alpha_1\alpha_0 = 11 \rightarrow U$

orward: Forward the input to the output.

oad "Register": Load the input into "Register".

lip: Clip input based on value stored in R.

erage 4: Average input with last 3 values stored in S, T, U.

fter executing each instruction, the controller nables the instruction counter to increment by ne.

RTL code

Forward:

```
R ← input;
output ← shiftLeft(R, 0);
```

Load X:

```
X ← input;
```

where X is R, S, T or U.

Clip:

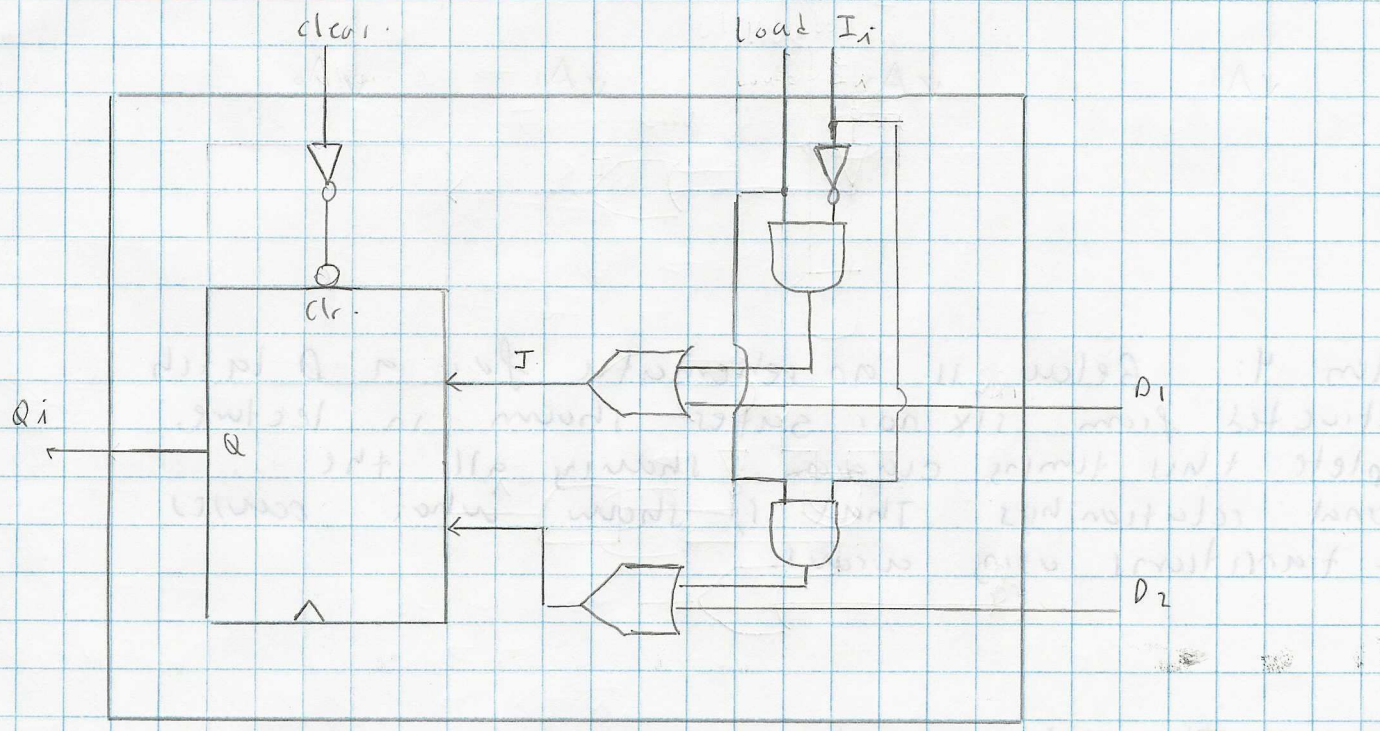
```
S ← input
if (S > R)
{
    U ← R;
}
else
{
    U ← S;
}
output ← shiftLeft(U, 0);
```

Increase 4:

```
R ← Input + U;
R ← R + T;
R ← R + S;
U ← T;
T ← S;
S ← input
```

Problem 3: Construct a 4 bit binary counter with parallel load using the JK negative edge triggered flip flop shown. The counter should have clear, Load, and count, a 4 bit data input I_{3-0} and a 4 bit data output A_{3-0} and a carry out.

First we create a bigger component with a JK flipflop inside that contains the clear and load logic

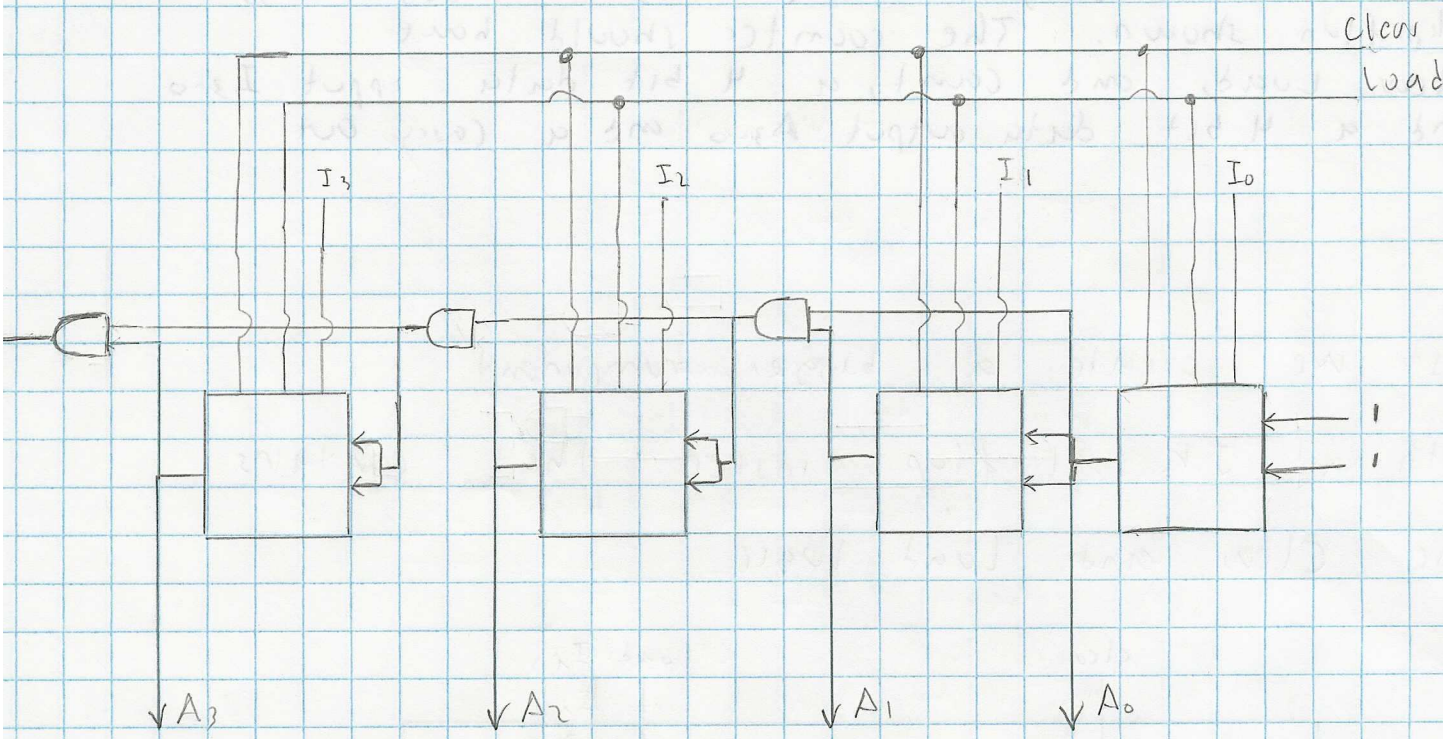


$$J = D_1 + (Load \cdot \bar{I}_i)$$

$$K = D_2 + (Load \cdot I_i)$$

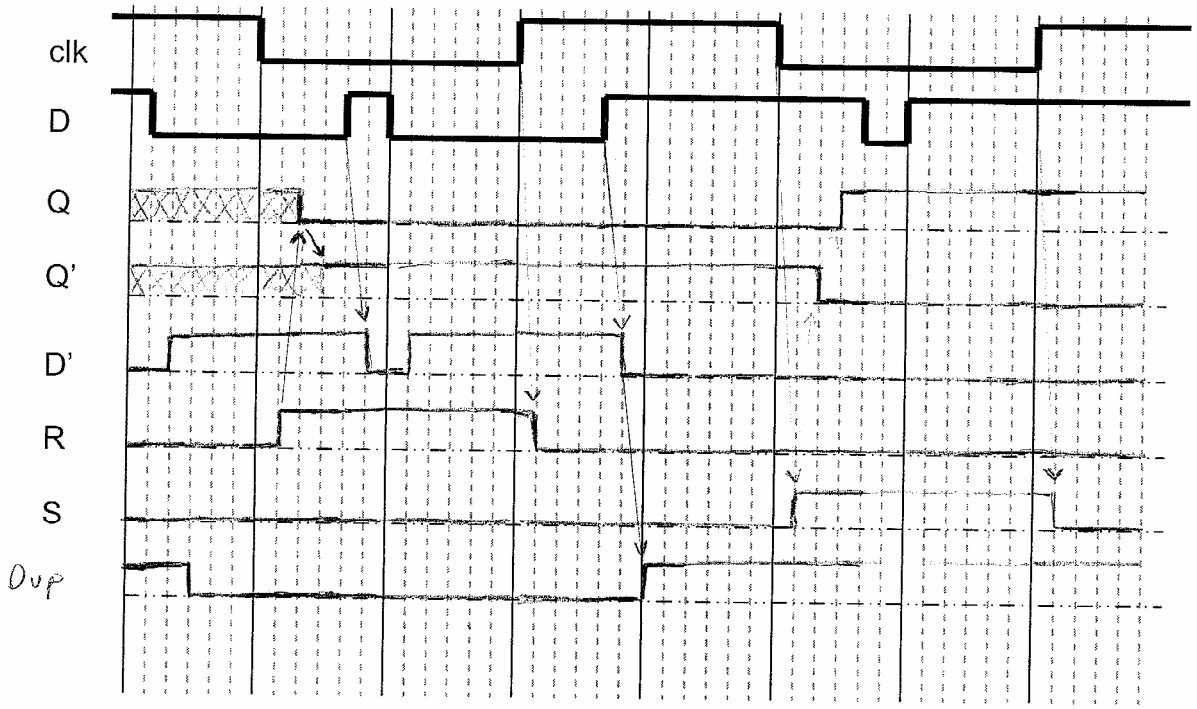
ow we connect 4 of these components

n the following way:



blem 4: Below is an schematic for a D latch nstructed from six nor gates shown in lecture. mplete this timing diagram showing all the tional relationships. That is, show what causes he transitions using arcs.

The timing diagram on the next page:


clk
D
Q
Q'
D'
R
S
Ovp