Lab 4: IP Design

Digital Design Lab

Overview

- IP (Intellectual Property) design, is to design a reusable unit of logic.
  - Methodology
  - Example: DCT
    - Lab procedure walkthrough

Methodology

Behavior:

1. Understand the functionality of DCT.
2. Create behavior using for-loops.
3. Create a testbench
4. Test the behavioral model.

Structure:

5. Understand the datapath components.
6. Refine computation according to datapath.
7. Create a timing table for registers.
8. Connect the datapath components.
9. Create FSMD.
10. Derive control word for controller.
11. Describe controller with VHDL processes.
12. Describe datapath as a set of connected components and connects it to controller.
13. Test the structural model.
14. Evaluate performance

1. Understand DCT

- **DCT** (Discrete Cosine Transform) is an important step in video/audio compression.
- ***Formula: Outblock*** $= COS_1 \times$ ***Inblock*** $\times COS_2$
- where, ***Outblock*** is 8x8 output matrix
- ***Inblock*** is 8x8 input matrix
- $COS_2 = COS_1^T$

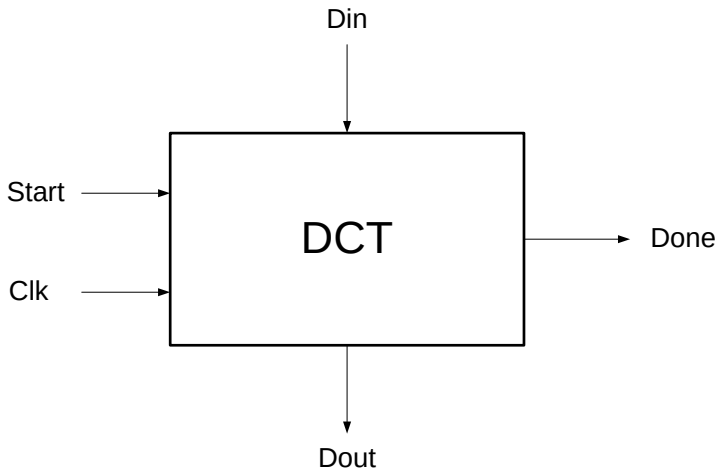
COS_1 is pre calculated as constant:

```
( 125,  122,   115,  103,   88,   69,   47,   24 ),
( 125,  103,    47,  -24,  -88, -122, -115, - 69 ),
( 125,   69,   -47, -122,  -88,   24,  115,  103 ),
( 125,   24,  -115,  -69,   88,  103,  -47, -122 ),
( 125,  -24,  -115,   69,   88, -103,  -47,  122 ),
( 125,  -69,   -47,  122,  -88,  -24,  115, -103 ),
( 125, -103,    47,   24,  -88,  122, -115,   69 ),
( 125, -122,   115, -103,   88,  -69,   47,  -24 )
```

Design Specification

- $Outblock = COS_1 \times Inblock \times COS_2$

- **Goal**: Design an IP core performing 8x8 DCT
  - Input: Start (1 bit), Din (32 bits) and Clk (1 bit)
  - Output: Done (1 bit) and Dout (32 bits)



C Reference Code

```
1. int COS1[8][8], COS2[8][8];
2. void MatrixMult(int a[][8], int b[][8], int c[][8])
3. { register int i, j, k;                    //c=a*b
4.    for (i=0; i<8; i++)
5.       for (j=0; j<8; j++)
6.       { c[i][j] = 0;
7.          for (k=0; k<8; k++)
8.          c[i][j] += a[i][k] * b[k][j];  }
9.   }
10.void DCT (int f[][8], int F[][8])
11.{ int Temp[8][8];
12.    MatrixMult(f, COS2, Temp);  //Temp=f*COS2
13.    MatrixMult(COS1, Temp, F); }//F=COS1*Temp
•                                   //equivalent as F=COS1*f*COS2
```

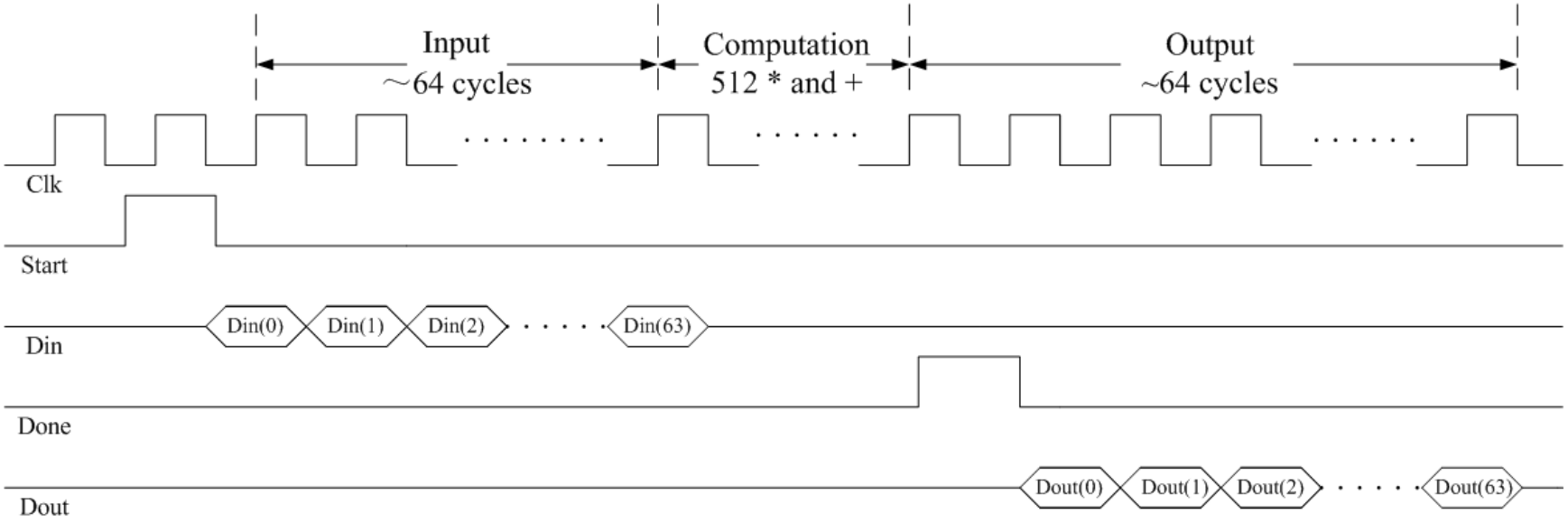
2. Create Behavior Using for-loops

```
1.  //for loops in C
2.  for (i=0; i<8; i++)
3.  {
4.    for (j=0; j<8; j++)
5.    {
6.        c[i][j] = 0;
7.        for (k=0; k<8; k++)
8.          c[i][j] += a[i][k] * b[k][j];
9.    }
10. }
```

```
1.  -- for loops in VHDL
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.
5.      BlockC( i, j ) := 0;
6.      for k in 0 to 7 loop
7.        BlockC( i, j ) = BlockC( i, j )
          + BlockA( i, k )*BlockB(k,j);
8.      end loop;
9.    end loop;
10. end loop;
```

Input, Computation and Output

- Input: After *start* is 1, load an 8x8 matrix from *Din*.
  - load in 64 integers one per cycle
- Computation: Matrix multipliation
  - Outblock = $COS_1 \times$ Inblock $\times$ COS_2 , where $COS_2 = COS_1^T$
- Output: After *done* is 1, output *an* 8x8 matrix from *Dout.*
  - output 64 integers one per cycle



DCT Behavior: Input

When start = '1', read in 64 integers one at a cycle

```
1.  --------------------------------
2.  -- Starting
3.  --------------------------------
4.  wait until Start = '1';
5.  Done <= '0';
6.  --------------------------------
7.  -- Read Input Data
8.  --------------------------------
9.  for i in 0 to 7 loop
10.   for j in 0 to 7 loop
11.     wait until Clk = '1' and Clk'event;
12.     InBlock( i, j ) := Din;
13.   end loop;
14. end loop;
```

DCT Behavior: Computation

- $\text{Outblock} = \text{COS}_1 \times \text{Inblock} \times \text{COS}_1^{\text{T}}$
- compute as two matrix multiplications:
- 1. $\text{Tempblock} = \text{COS}_1 \times \text{Inblock}$ 2. $\text{Outblock} = \text{Tempblock} \times \text{COS}_1^{\text{T}}$

```
-- TempBlock = COSBlock * InBlock
  for i in 0 to 7 loop
    for j in 0 to 7 loop
           TempBlock( i, j ) := 0;
       for k in 0 to 7 loop
         TempBlock(i,j):= TempBlock(i,j)
              +COSBlock(i,k)* InBlock(k,j);
          end loop;
      end loop;
   end loop;
```

```
-- OutBlock = TempBlock * COS1T
  for i in 0 to 7 loop
    for j in 0 to 7 loop
           OutBlock( i, j ) := 0;
       for k in 0 to 7 loop
         OutBlock(i,j):= OutBlock(i,j)
              + TempBlock(i,k)*COSBlock(j,k);
          end loop;
      end loop;
   end loop;
```

DCT Behavior: Output

When done = ' 1', write out 64 integers one at cycle

```
1.   ---------------------------------
2.   -- Finishing
3.   ---------------------------------
4.   wait until Clk = '1' and Clk'event;
5.   Done <= '1';
6.   ---------------------------------
7.   -- Output Data
8.   ---------------------------------
9.   for i in 0 to 7 loop
10.    for j in 0 to 7 loop
11.      wait until Clk = '1' and Clk'event;
12.      Done <= '0';
13.      Dout <= OutBlock( i, j );
14.    end loop;
15.  end loop;
```

The Whole Picture

```
1.  -- Starting
2.    wait until Start = '1';
3.    Done <= '0';

4.  --- Read Input Data
5.    for i in 0 to 7 loop
6.      for j in 0 to 7 loop
7.        wait until Clk = '1'
            and Clk'event;
8.        BlockB( i, j ) := Din;
9.      end loop;
10.   end loop;

11. ----Computing: matrix multiplication 1
12. …
13. ----Computing: matrix multiplication 2
14. …
15. ---Finishing
16.   wait until Clk = '1' and Clk'event;
17.   Done <= '1';
18.
19. --Output Data
20.   for i in 0 to 7 loop
21.     for j in 0 to 7 loop
22.       wait until Clk = '1' and
    Clk'event;
23.       Done <= '0';
24.       Dout <= BlockC( i, j );
25.     end loop;
26.    end loop;
27.  end process;
28.
```

3A. Create Testbench

- One test input:

255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

- Correct (expected) Output:

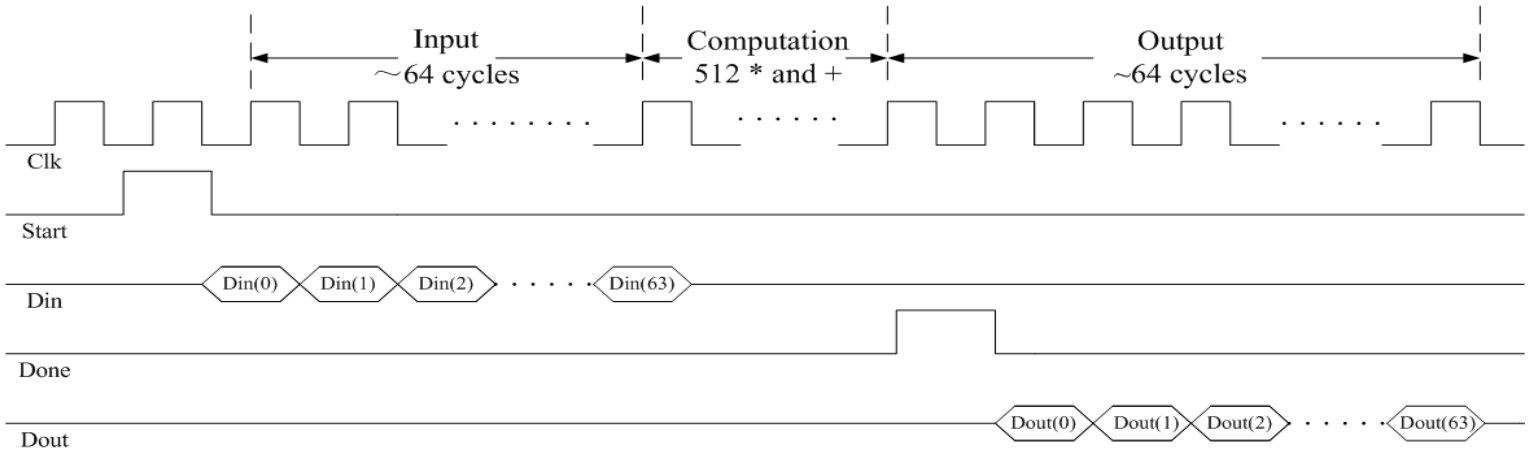
88710930	-18305430	22913790	-1664130	14721150	3968310	10368810	7296570
-5478165	1130415	-1414995	102765	-909075	-245055	-640305	-450585
68742135	-14184885	17755905	-1289535	11407425	3075045	8034795	5654115
-5654880	1166880	-1460640	106080	-938400	-252960	-660960	-465120
-7422030	1531530	-1917090	139230	-1231650	-332010	-867510	-610470
-530145	109395	-136935	9945	-87975	-23715	-61965	-43605
25623675	-5287425	6618525	-480675	4252125	1146225	2994975	2107575
12723480	-2625480	3286440	-238680	2111400	569160	1487160	1046520

3B. Create Testbench

```
1.  …
2.  --Starting
3.  Start =  '1';
4.  --Feed Input
5.  for i in 0 to 23 loop
6.    wait until Clk = '1' AND Clk'EVENT;
7.    Din <= 255;
8.    Start =  '0';
9.  end loop;
10. for i in 24 to 39 loop
11.   wait until Clk = '1' AND Clk'EVENT;
12.   Din <= 0;
13. end loop;
14. for i in 40 to 63 loop
15.   wait until Clk = '1' AND Clk'EVENT;
16.   Din <= 255;
17. end loop;
18. --Finishing
19.   wait until Done = '1';
20. wait until Clk = '1' AND Clk'EVENT;
21. …
```

4. Test the Behavioral Model

- Expected waveform



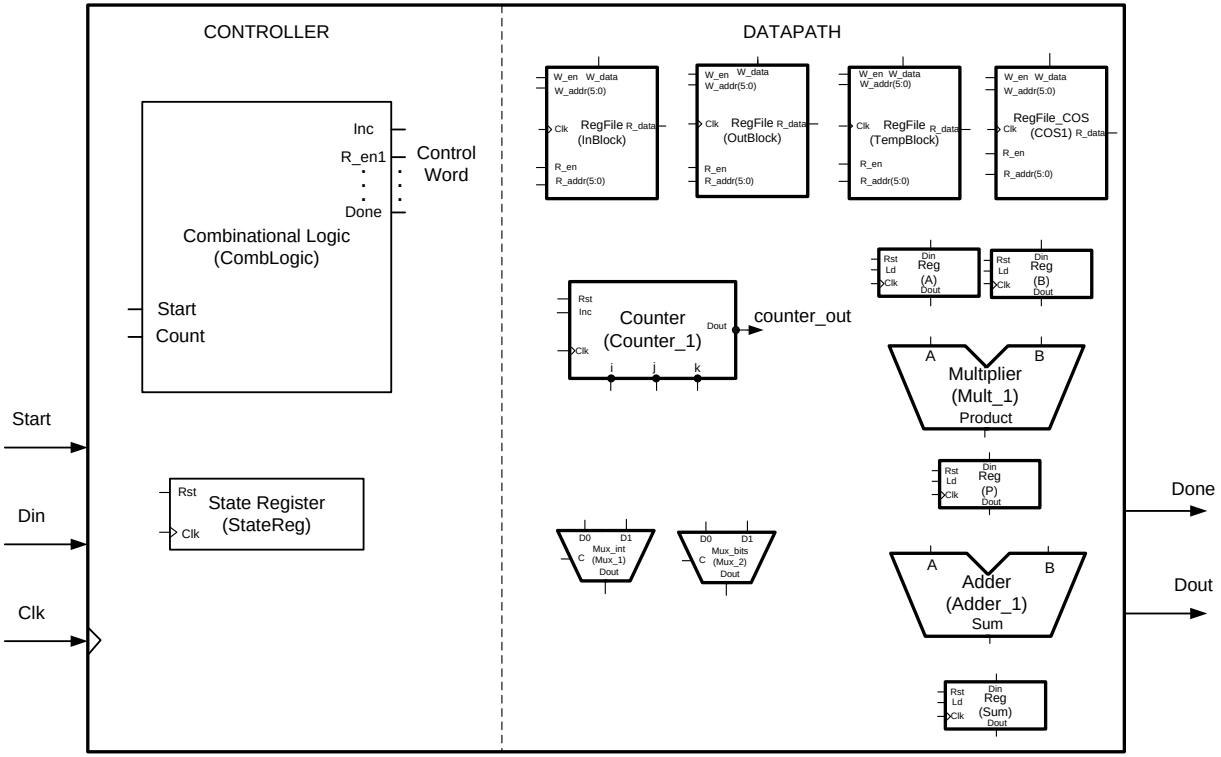
- Expected output:

```
88710930 -18305430 22913790 -1664130 14721150 3968310 10368810 7296570
-5478165 1130415 -1414995 102765 -909075 -245055 -640305 -450585
68742135 -14184885 17755905 -1289535 11407425 3075045 8034795 5654115
-5654880 1166880 -1460640 106080 -938400 -252960 -660960 -465120
-7422030 1531530 -1917090 139230 -1231650 -332010 -867510 -610470
-530145 109395 -136935 9945 -87975 -23715 -61965 -43605
25623675 -5287425 6618525 -480675 4252125 1146225 2994975 2107575
12723480 -2625480 3286440 -238680 2111400 569160 1487160 1046520
```

- See Xilinx window.

5. Understand Datapath Components

- Control Word:
- Inc,
- R_en1,
- R_en2,
- R_en3,
- R_en4 ,
- LoadSum,
- Rst_counter ,
- Rst_sum ,
- W_en1 ,
- W_en2 ,
- W_en3 ,
- W_en4 ,
- Sel1 ,
- Sel2



6. Refine Computation according to DP

- Long equation requires multiple clock cycles to compute.
- Rewrite computation into multi-clock steps using intermediate variables A, B, P and Sum.
- Each variable is stored in a register.

```
1.  -- TempBlock = COSBlock * InBlock
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      TempBlock( i, j ) := 0;
5.      for k in 0 to 7 loop
6.        TempBlock( i, j ) := TempBlock( i,j )
7.               +COSBlock( i,k )*InBlock( k, j );
8.      end loop;
9.    end loop;
10. end loop;
```

```
1.  -- TempBlock = COSBlock * InBlock
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      Sum := 0;
5.      for k in 0 to 7 loop
6.        A := COSBlock( i, k );
7.        B := InBlock( k, j );
8.        P := A * B;
9.        Sum := Sum + P;
10.       if( k = 7 ) then
11.         TempBlock( i, j ) := Sum;
12.       end if;
13.     end loop;
14.   end loop;
15. end loop;
```

7. Create Timing Table for Registers

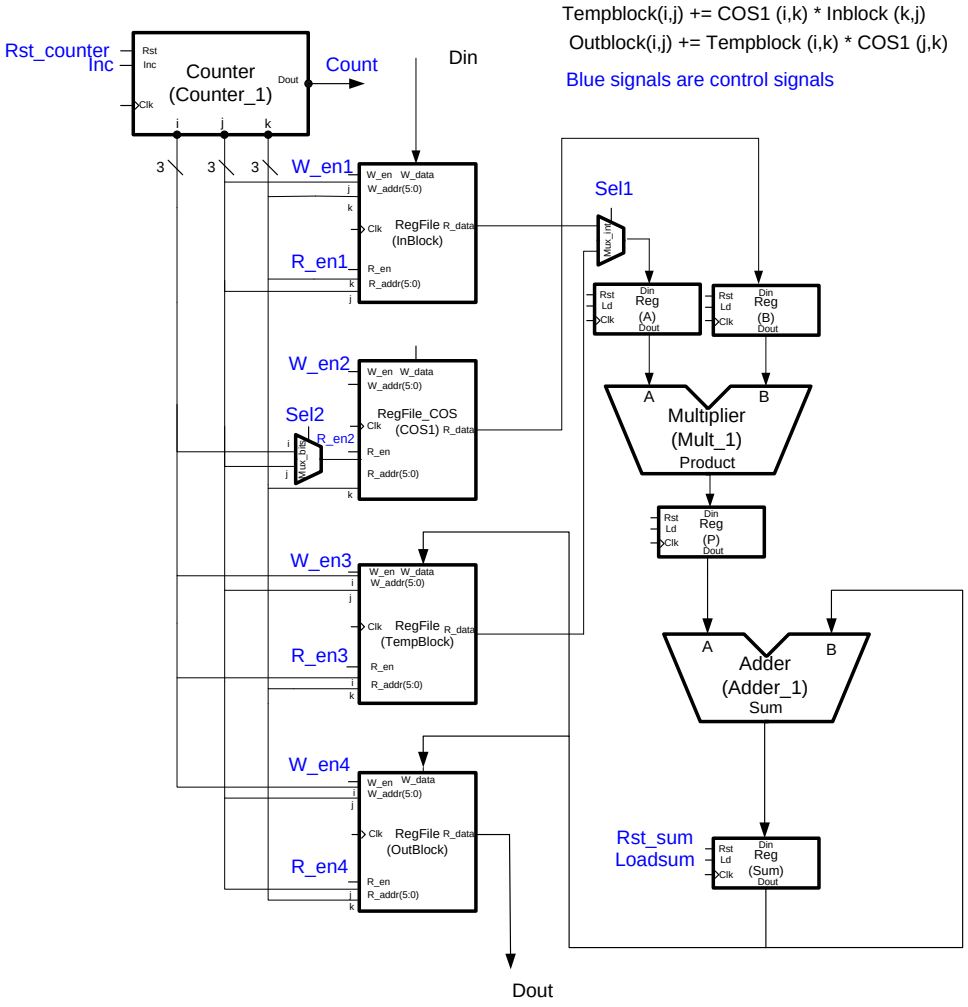
- Build a timing table indicating the value in each register in each clock cycle.

```
1.  -- TempBlock = COSBlock * InBlock
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      Sum := 0;
5.      for k in 0 to 7 loop
6.        A := COSBlock( i, k );
7.        B := InBlock( k, j );
8.        P := A * B;
9.        Sum := Sum + P;
10.       if( k = 7 ) then
11.         TempBlock( i, j ) := Sum;
12.       end if;
13.     end loop;
14.   end loop;
15. end loop;
```

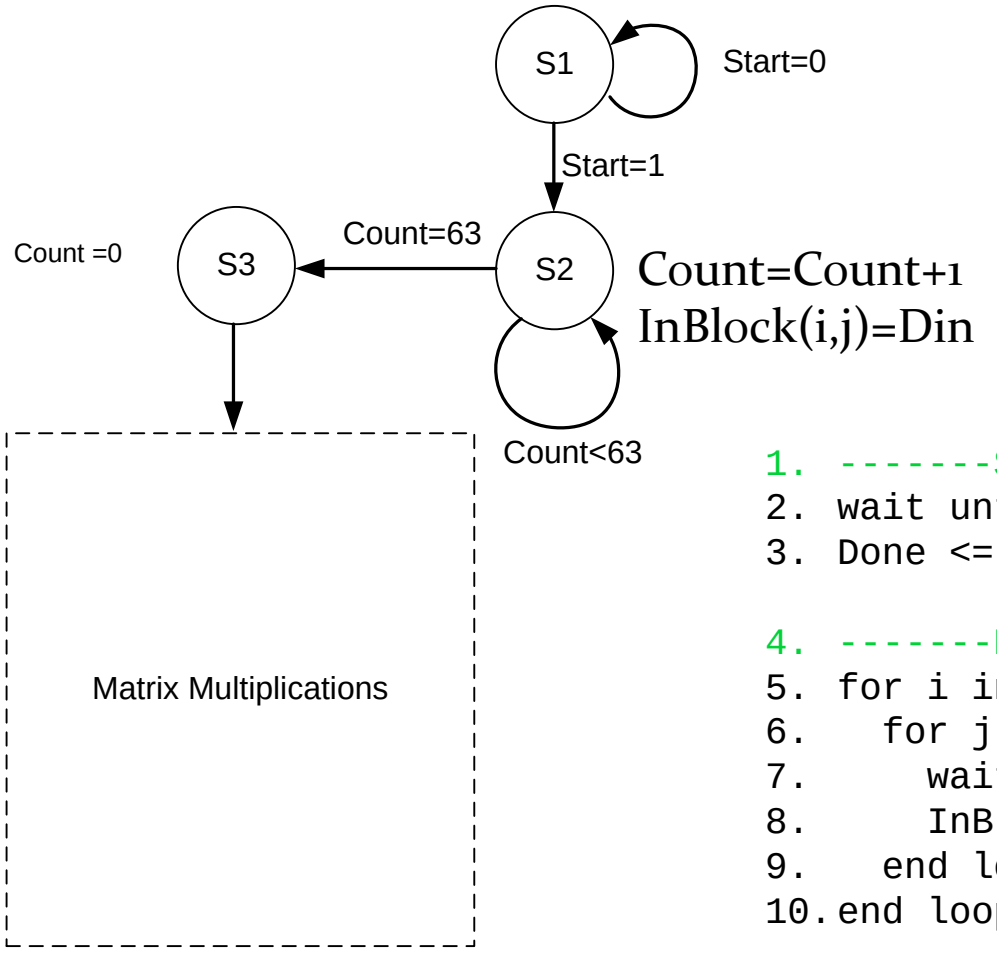
Cyc# / Regs	1	2	3	4	5	6
A,B	Fetch A,B			Fetch A,B		
P		A*B			A*B	
Sum			P+Sum			P+Sum

9. Connect Datapath Components

- Blue signals are connections with Controller.
- Counter is used to address RegFiles and provide status signal to Controller.
- Muxes is used to switch between input data needed by two matrix mulplications
- Connection done by port mapping in VHDL

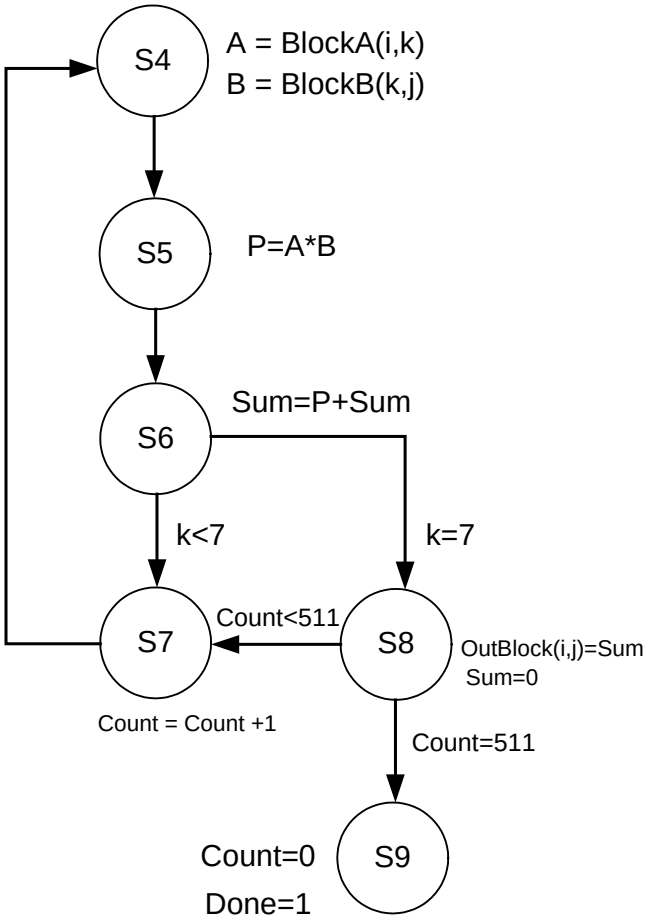


10A. Create FSMD for Input



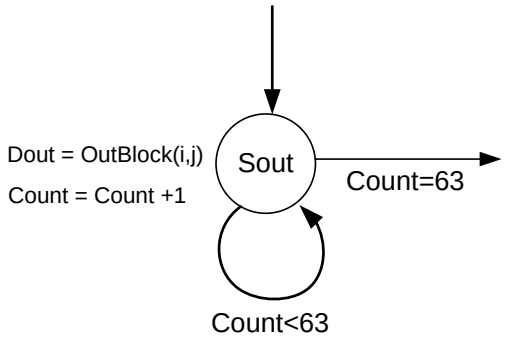

```
1.  -------Starting
2.  wait until Start = '1';
3.  Done <= '0';

4.  -------Read Input Data
5.  for i in 0 to 7 loop
6.    for j in 0 to 7 loop
7.      wait until Clk = '1' and Clk'event;
8.      InBlock( i, j ) := Din;
9.    end loop;
10. end loop;
```

10B. Create FSMD for Computation



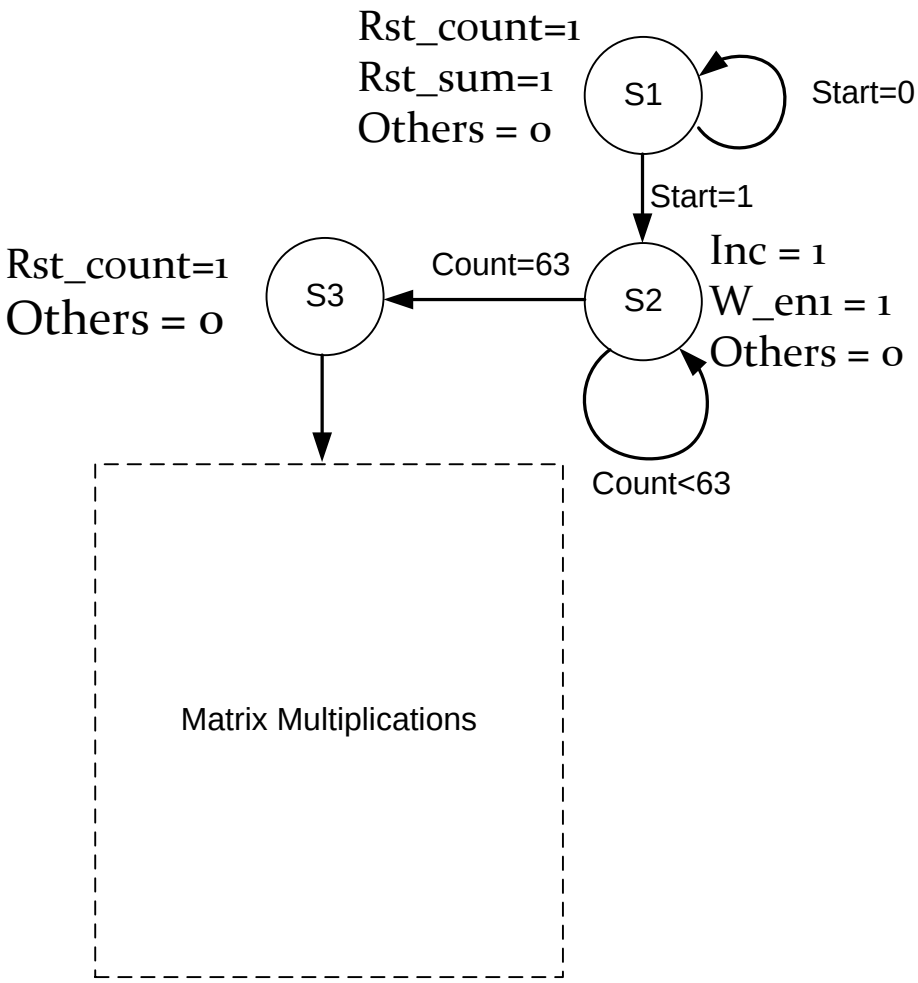

```
1.  --------- Matrix Multiplication
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      Sum := 0;
5.      for k in 0 to 7 loop
6.        A := BlockA( i, k );
7.        B := BlockB( k, j );
8.        P := A * B;
9.        Sum := Sum + P;
10.       if( k = 7 ) then
11.         OutBlock( i, j ) := Sum;
12.       end if;
13.     end loop;
14.   end loop;
15. end loop;
```

10C. Create FSMD for Output



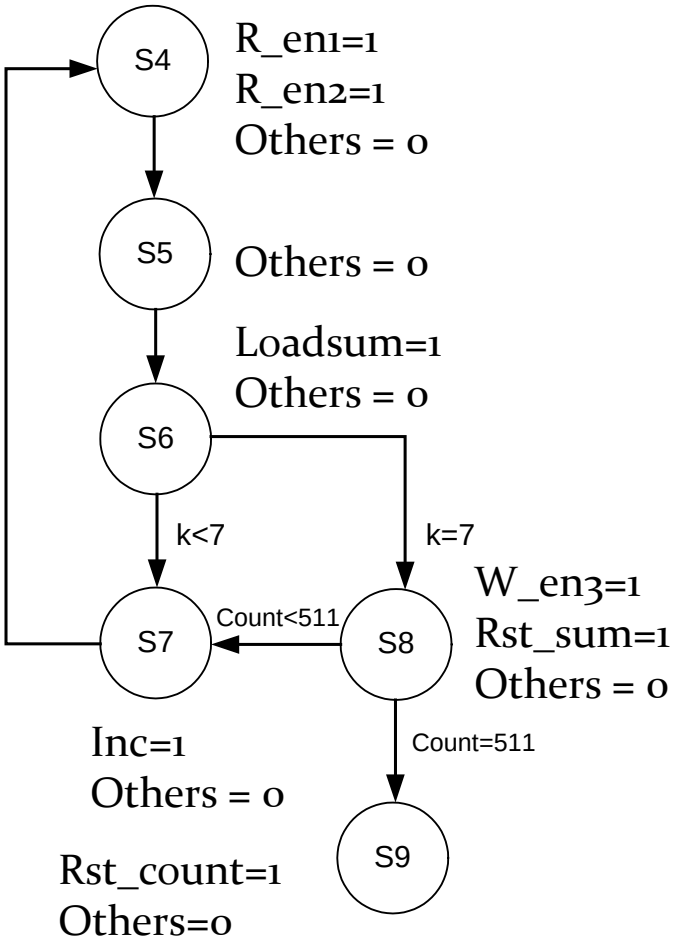

```
1.  ------------Output Data
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      wait until Clk = '1' and Clk'event;
5.      Done <= '0';
6.      Dout <= OutBlock( i, j );
7.    end loop;
8.  end loop;
9.  end process;
```

11A. Derive Control Word for Input



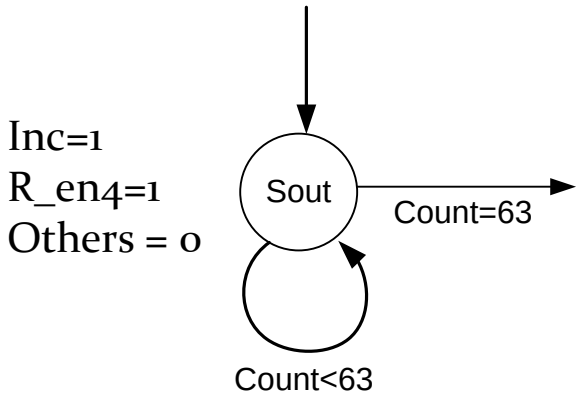

```
1.  -------Starting
2.  wait until Start = '1';
3.  Done <= '0';

4.  -------Read Input Data
5.  for i in 0 to 7 loop
6.    for j in 0 to 7 loop
7.      wait until Clk = '1' and Clk'event;
8.      BlockB( i, j ) := Din;
9.    end loop;
10. end loop;
```

11B. Derive Control Word for Computation



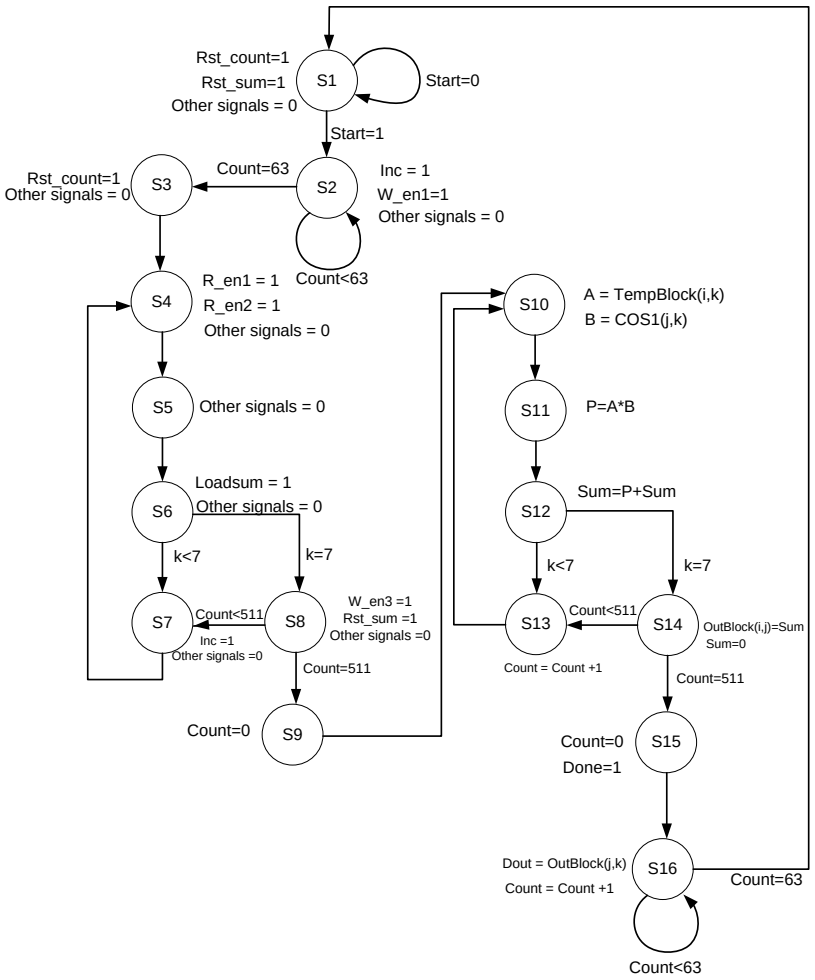

```
1.  --------- Matrix Multiplication
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      Sum := 0;
5.      for k in 0 to 7 loop
6.        A := BlockA( i, k );
7.        B := BlockB( k, j );
8.        P := A * B;
9.        Sum := Sum + P;
10.       if( k = 7 ) then
11.         BlockC( i, j ) := Sum;
12.       end if;
13.     end loop;
14.   end loop;
15. end loop;
```

11C. Derive Control Word for Output

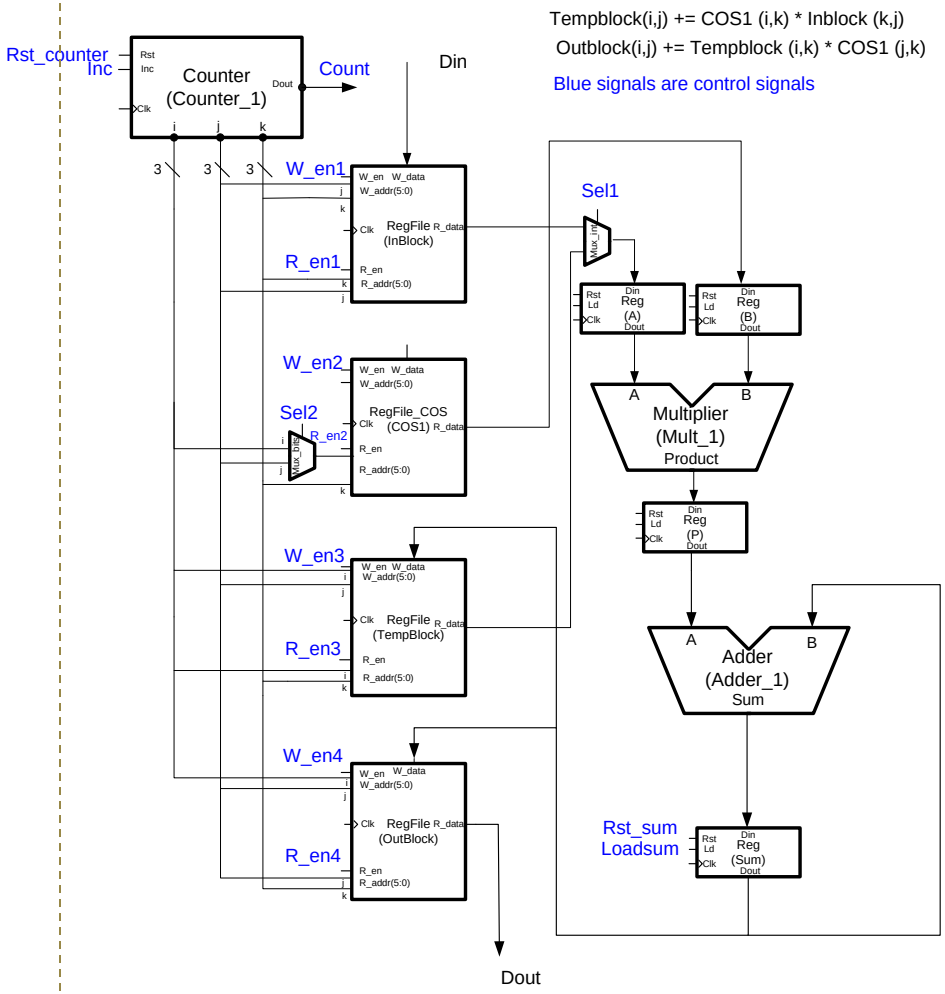



```
1.  ------------Output Data
2.  for i in 0 to 7 loop
3.    for j in 0 to 7 loop
4.      wait until Clk = '1' and Clk'event;
5.      Done <= '0';
6.      Dout <= BlockC( i, j );
7.    end loop;
8.  end loop;
9.  end process;
```

The Complete Controller + Datapath



Controller



Datapath

12-13.

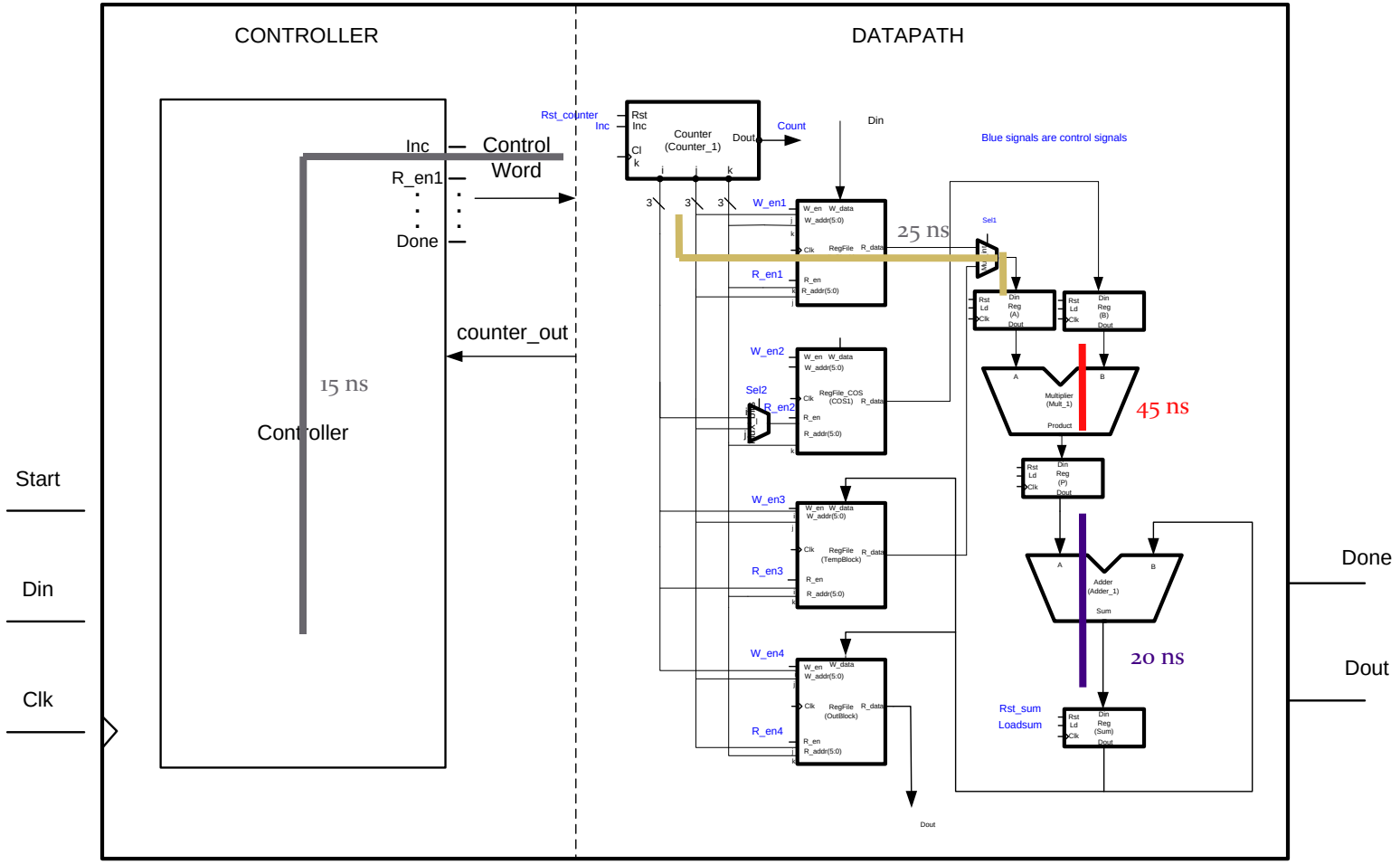
- 12. Describe controller with 2 processes: State Register and Combinational Logic.
  - Translate the controller FSM diagram into VHDL case statement
- 13. Describe datapath as a set of connected components and connects it to controller
  - Port mapping

14. Test the structure with testbench

- Test the structural model with the testbench.
  - Use same testbench as in behavior model
  - See Xilinx ISE window for results

15. Evaluate Performance

- Minimal Clock Cycle = 45 ns
- Execution time ≈ 4300 cycles



Summary

- Though the procedure, we have
  - Created DCT behavioral model
  - Synthesized from behavior to register-transfer level structural model