William Moffett
wcm160130

Architecture Design

Modules

MSDAP – contains the finite state machine that drives FIR module

How this will work will be described in the following specification.

IO – manages data input in FIR module

This module will collect data from the input of the MSDAP module and gather the data every Dclk cycle. Once all 16 bits have been gathered, the IO module will generate a valid signal so that the FIR module can collect the 16 bits of data and perform its computation on it. The IO module will also count the amount of data values the FIR module has received to inform the MSDAP module when to change states. The IO module will also count the amount of times a datum of 0 has been given to the FIR modules. Once this count reaches 800, the IO module will disable the clock of the FIR module to save power.

FIR – performs the finite impulse response computation

There are 2 FIR modules in the MSDAP module for 2 streams of data. The FIR will execute the MSDAP algorithm in less than 560 clock cycles and then generate a valid signal to communicate that the data in its output port is valid. This directly informs the MSDAP module when to put its OutReady port high.

Specification

MSDAP IO's

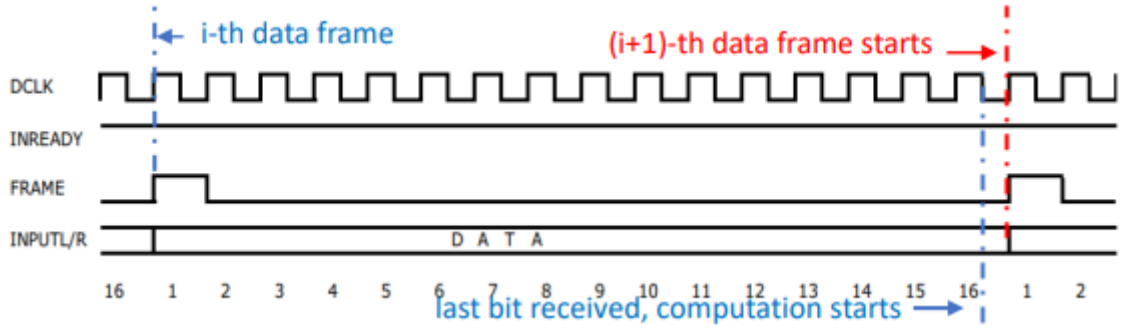
Inputs

Sclk – System clock
Drives computation
Dclk – Data clock
Drives data input / output
Start – Start signal
Starts chip when set low
Reset – Reset signal
Resets chip when set high
Frame – Aligned with Dclk
Informs chip to start listening for incoming data
InputL – Data input for channel left
Chip reads from this channel after receiving Frame high
InputR – Data input for channel right
Chip reads from this channel after receiving Frame high

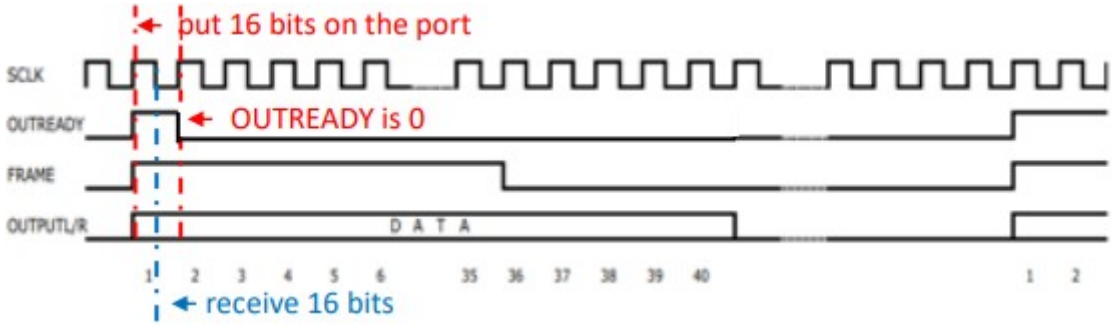
Outputs

InReady – Output signal
Informs controller chip is ready to receive input
OutReady – Output signal
Informs controller output is ready
OutputL – Output data for channel left
Controller reads data from this channel
OutputR – Output data for channel right
Controller reads data from this channel

Every input/output signal is 1 bit except for OutputL and OutputR which are 40 bits each. Input is read one bit at a time on the InputL and InputR channels. Input is read on the negative edge of the Dclk after Frame is set high.

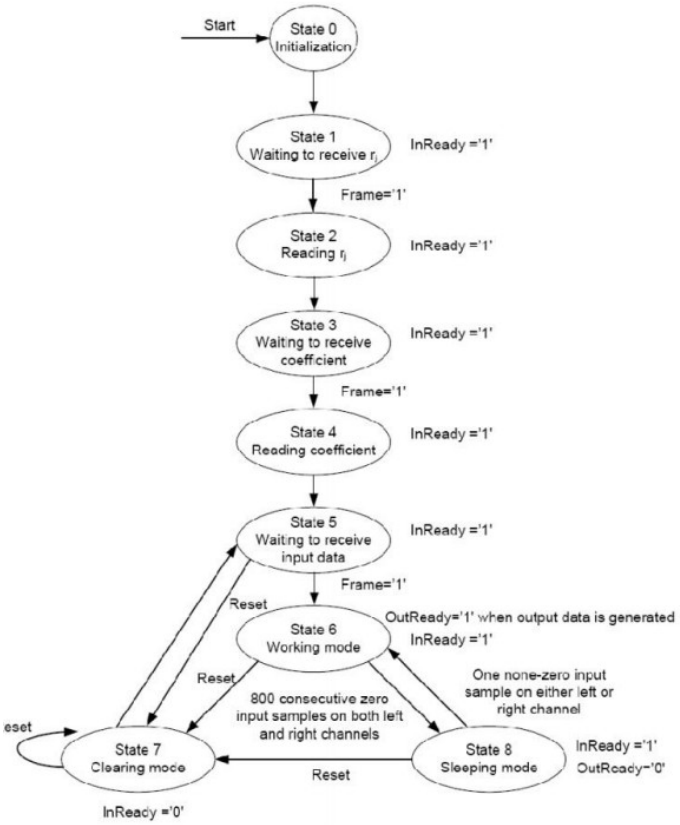


OutputL and OutputR are 40 bits and can be read in 1 Dclk cycle. The controller reads the outputs on the negative edge of the Dclk after receiving OutReady high.



Operation Modes

The MSDAP chip contains a finite state machine that controls the operation of the chip. Here is a diagram that shows the transitions between the states.



When the chip receives Start low, it transitions to state 1. It is ready to receive Rj values. In state 2 it reads the Rj values. After it has read 16 Rj values, it goes to state 3, where it is ready to read coefficients. In state 4 it reads the coefficients. After 512 coefficients have been read it goes to state 5, where it is ready to read in data. After it has read one datum, it goes to state 6 where it starts computing values. If in state 6 it receives a reset signal, it goes to state 7 where it resets the computation. At the end of the reset it goes back to state 5. If in state 6 the chip reads 800 zeroes it goes to state 8 where the Sclk is disabled for the computation. After the chip reads a nonzero value, it returns to state 6. If a reset is detected while in state 8, it will go to state 7.

IO format

The start signal is received when set low.

The reset signal is received when set high.

The frame signal is received when set high.

Input data is received in serial, each bit is received by the chip on the negative edge of the Dclk.

Output data is received in parallel, it is received by the controller after OutReady is set high.

Frequency of Sclk

The frequency of Sclk is determined by the frequency of Dclk. Sclk must be atleast 35 times faster than Dclk. Setting the frequency of Sclk too high may result in errors.

Module Specifications

MSDAP – implements previous spec, converts start low into start high for other submodules

IO -
Inputs

clk	1 bit	Module receives Dclk through this port
start	1 bit	Module receives start (high) through this port
rst	1 bit	Module receives rst (high) through this port
rstcount	1 bit	On high, module will reset its internal count registers
frame	1 bit	Module receives frame input through this port
inL	1 bit	Module receives left input channel through this port
inR	1 bit	Module receives right input channel through this port

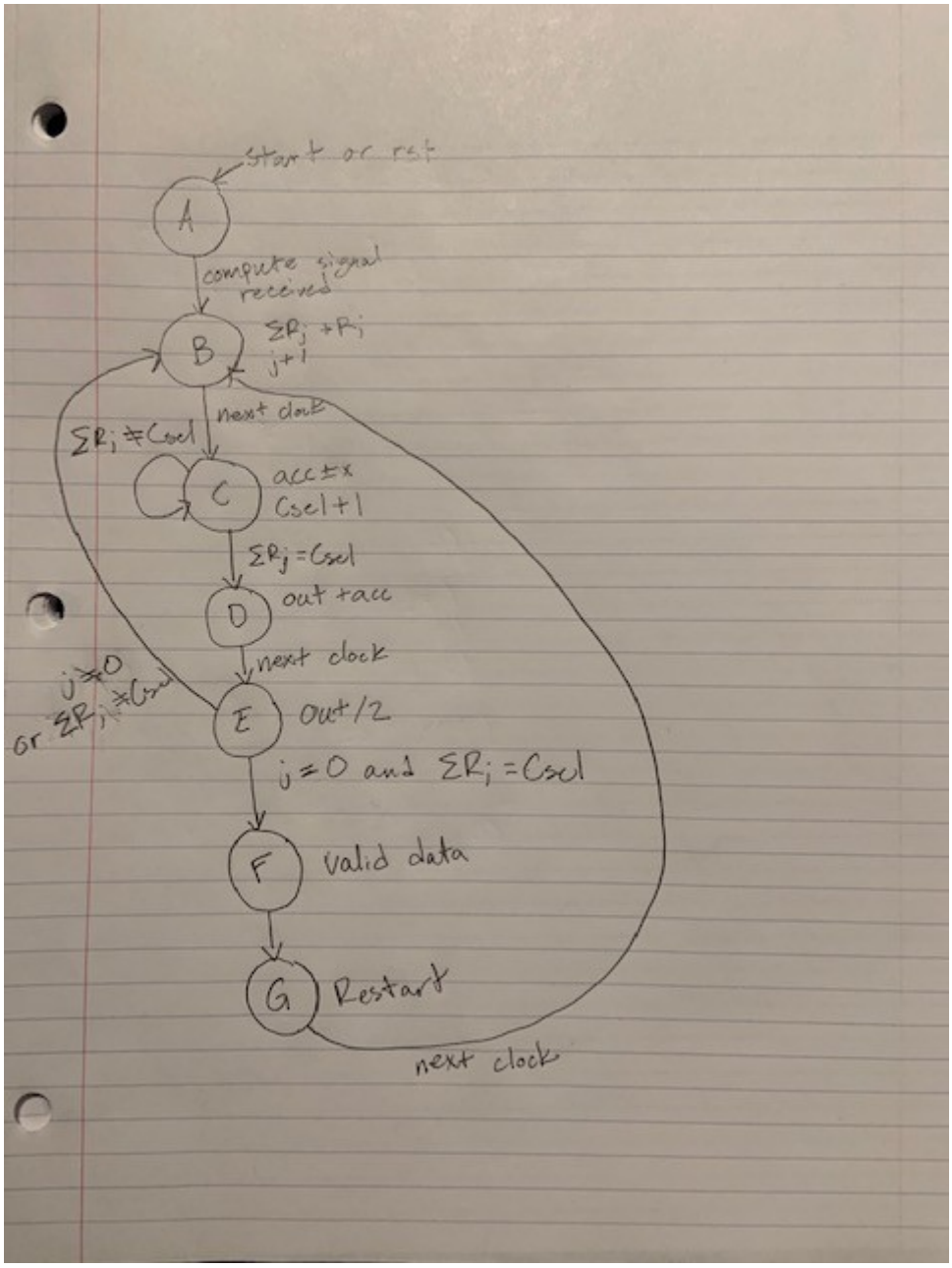
Outputs

outL	16 bits	Data input into left FIR module
outR	16 bits	Data input into right FIR module
incount	10 bits	Counts the amount of inputs into FIR modules
datavalid	1 bit	Set high when outL and outR are valid for FIR module
disableclk	1 bit	Set high when 800 zeroes are read from inL and inR
hasdata	1 bit	Set high when FIR module has been sent atleast 1 valid datum

FIR

x	16 bits	Input data for computation
datain	1 bit	Valid signal for x port
clk	1 bit	Module receives Sclk through this port
rst	1 bit	Module receives rst (high) through this port
start	1 bit	Module receives start (high) through this port
comp	1 bit	When set high, enables FSM
out	40 bit	Output of computation
dataout	1 bit	Valid signal for out port

FIR FSM



Verification of Design

For Loops in FIR Shift Register

In the shift register of the FIR module there are 3 for loops. 2 for loops are used to reset the values in the register to zero. To do this without a for loop would require an excessive amount of typing to achieve the same result. 1 for loop shifts all the values in the register down 1 index. This, again, would require an excessive amount of typing to achieve the same result. The design compiler probably has a specific edge case to detect when these behaviors are desired, and there appears to be no downside to programming this functionality this way.

Changing Code to be Synthesizable

The main difficulty with changing the code was implementing single drivers for each register. No register can be driven by multiple clocks, nor can they be driven by both the positive and negative edge of a clock. This challenge was difficult to overcome at first, but once a strategy is developed to avoid these issues, designing is not too difficult. Once synthesizable code was made, making work correctly is another story.

Correctness

Comparing generated RTL values with correct outputs.

```
00e13599c4 0014a70466 | //   5569      00E13599C4      0014A70466

ffb99a48bd ffb815dd4d | //   5570      FFB99A48BD      FFB815DD4D

fecfe222ac 0032eb97f8 | //   5571      FECFE222AC      0032EB97F8

001df59768 fff0acb492 | //   5572      001DF59768      FFF0ACB492

ff5c63ab5b ffbf98bbbb | //   5573      FF5C63AB5B      FFBF98BBBB

00899b6e46 ffdb793db2 | //   5574      00899B6E46      FFDB793DB2

006eea6a3d ff831698a4 | //   5575      006EEA6A3D      FF831698A4
```

Comparing generated netlist values with correct outputs.

```
00318d9f23 007ff01b98 | //   5437        00318D9F23      007FF01B98
ffeb0e2223 002540decf | //   5438        FFEB0E2223      002540DECF
001df803d2 000356f3f7 | //   5439        001DF803D2      000356F3F7
ffad530583 ffc15a433e | //   5440        FFAD530583      FFC15A433E
ffdc012903 000db34f04 | //   5441        FFDC012903      000DB34F04
00006c8600 002a29678e | //   5442        00006C8600      002A29678E
000f8c4dd8 ff7b2c1ee8 | //   5443        000F8C4DD8      FF7B2C1EE8
003a07fe90 00659c8d0e | //   5444        003A07FE90      00659C8D0E
ffeadc62be 004032c4ba | //   5445        FFEADC62BE      004032C4BA
```

Both the RTL and netlist generated all the correct values.

Area Report

```
****************************************
Report : area
Design : MSDAP
Version: L-2016.03-SP3
Date   : Tue Dec  7 16:45:20 2021
****************************************

Library(s) Used:

    ss_1v62_125c (File: /home/eng/y/yxw173630/lib/synopsys/ss_1v62_125c.db)

Number of ports:                 2191
Number of nets:                 69819
Number of cells:                59465
Number of combinational cells:  41345
Number of sequential cells:     18074
Number of macros/black boxes:       0
Number of buf/inv:               8797
Number of references:              31

Combinational area:       520411.665733
Buf/Inv area:              74935.346054
Noncombinational area:    805899.632864
Macro/Black Box area:          0.000000
Net Interconnect area:    undefined  (No wire load specified)

Total cell area:         1326311.298597
Total area:                undefined
1
```

Cell Report

```
****************************************
Report : cell
Design : MSDAP
Version: L-2016.03-SP3
Date   : Tue Dec  7 16:45:40 2021
****************************************

Attributes:
    b - black box (unknown)
    h - hierarchical
    n - noncombinational
    r - removable
    u - contains unmapped logic

Cell                 Reference      Library          Area  Attributes
------------------------------------------------------------------------------
OutReady_reg         DFFRX2M        ss_1v62_125c    50.489601 n
```

U36 AOI31X1M ss_1v62_125c 13.171200
U37 OAI31X1M ss_1v62_125c 13.171200
U38 INVX2M ss_1v62_125c 6.585600
U39 INVX2M ss_1v62_125c 6.585600
U40 NAND3BX2M ss_1v62_125c 13.171200
U41 INVX2M ss_1v62_125c 6.585600
U42 NOR2X2M ss_1v62_125c 8.780800
U43 NAND4BX1M ss_1v62_125c 15.366400
U44 OAI2B2X2M ss_1v62_125c 17.561600
U45 OAI22X1M ss_1v62_125c 13.171200
U46 NOR2X2M ss_1v62_125c 8.780800
U47 NOR4BX1M ss_1v62_125c 15.366400
U48 OAI22X1M ss_1v62_125c 13.171200
U49 NOR2BX2M ss_1v62_125c 10.976000
U50 AOI211X2M ss_1v62_125c 13.171200
U51 OAI2BB2X1M ss_1v62_125c 15.366400
U52 NAND4BBX1M ss_1v62_125c 19.756800
U53 NOR3X2M ss_1v62_125c 10.976000
U54 NOR4X1M ss_1v62_125c 13.171200
U55 OAI21X1M ss_1v62_125c 10.976000
U56 OAI21X1M ss_1v62_125c 10.976000
U57 NOR4BX1M ss_1v62_125c 15.366400
U58 OAI21X1M ss_1v62_125c 10.976000
U59 AOI32X1M ss_1v62_125c 15.366400
U60 NAND3X2M ss_1v62_125c 10.976000
U61 NOR2X2M ss_1v62_125c 8.780800
U62 NAND2X2M ss_1v62_125c 8.780800
U63 NAND2BX2M ss_1v62_125c 10.976000
U64 NOR2BX2M ss_1v62_125c 10.976000
comp_reg DFFRX2M ss_1v62_125c 50.489601 n
firL FIR_0 657444.836412
h, n
firR FIR_1 658175.838001
h, n
io IO 6063.142492
h, n
outdffL DFF40_0 1938.361647
h, n
outdffR DFF40_1 1938.361647
h, n
rst_reg TLATX1M ss_1v62_125c 28.537600 n
rstcount_reg DFFRQX2M ss_1v62_125c 46.099201 n
startdff DFF1_0 52.684801 h, n
state_reg[0] DFFSX2M ss_1v62_125c 50.489601 n
state_reg[1] DFFRX2M ss_1v62_125c 50.489601 n
state_reg[2] DFFSRX2M ss_1v62_125c 72.441597 n

Total 42 cells 1326311.298597
1

QOR Report

```
****************************************
Report : qor
Design : MSDAP
Version: L-2016.03-SP3
Date   : Tue Dec  7 16:45:50 2021
****************************************
```

Timing Path Group 'Dclk'

Levels of Logic:	9.00
Critical Path Length:	2.95
Critical Path Slack:	1.80
Critical Path Clk Period:	1302.00
Total Negative Slack:	0.00
No. of Violating Paths:	0.00
Worst Hold Violation:	0.00
Total Hold Violation:	0.00
No. of Hold Violations:	0.00

Timing Path Group 'Sclk'

Levels of Logic:	53.00
Critical Path Length:	14.75
Critical Path Slack:	0.00
Critical Path Clk Period:	30.00
Total Negative Slack:	0.00
No. of Violating Paths:	0.00
Worst Hold Violation:	0.00
Total Hold Violation:	0.00
No. of Hold Violations:	0.00

Cell Count

Hierarchical Cell Count:	46
Hierarchical Port Count:	2102
Leaf Cell Count:	59376
Buf/Inv Cell Count:	8797
Buf Cell Count:	7670
Inv Cell Count:	1127
CT Buf/Inv Cell Count:	2
Combinational Cell Count:	41345
Sequential Cell Count:	18031
Macro Count:	0

Area

Combinational Area: 520411.665733
Noncombinational Area:
805899.632864
Buf/Inv Area: 74935.346054
Total Buffer Area: 67456.30
Total Inverter Area: 7479.05
Macro/Black Box Area: 0.000000
Net Area: 0.000000

Cell Area: 1326311.298597
Design Area: 1326311.298597

Design Rules

Total Number of Nets: 67729
Nets With Violations: 0
Max Trans Violations: 0
Max Cap Violations: 0

Hostname: engnx05a.utdallas.edu

Compile CPU Statistics

Resource Sharing: 13.10
Logic Optimization: 37.71
Mapping Optimization: 100.63

Overall Compile Time: 163.81
Overall Compile Wall Clock Time: 166.78

Design WNS: 0.00 TNS: 0.00 Number of Violating Paths: 0

Design (Hold) WNS: 0.00 TNS: 0.00 Number of Violating Paths: 0

Resources Report

```
****************************************
Report : resources
Design : MSDAP
Version: L-2016.03-SP3
Date   : Tue Dec  7 16:46:14 2021
****************************************

Resource Sharing Report for design MSDAP in file
        /home/013/w/wc/wcm160130/MSDAPproj/MSDAP.v

================================================================================
========
|          |            |           | Contained     |                   |
| Resource | Module     | Parameters | Resources     | Contained Operations |
================================================================================
========
| r58      | DW01_cmp2  | width=4    |               | gt_20             |
================================================================================
========


No implementations to report
1
```

Timing Report

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : MSDAP
Version: L-2016.03-SP3
Date   : Tue Dec  7 16:45:00 2021
****************************************

Operating Conditions: ss_1v62_125c   Library: ss_1v62_125c
Wire Load Model Mode: top

  Startpoint: Start (input port clocked by Sclk)
  Endpoint: io/zerocounter_reg[9]
            (rising edge-triggered flip-flop clocked by Dclk)
  Path Group: Dclk
  Path Type: max

  Point                              Incr       Path
  -----------------------------------------------------------
  clock Sclk (rise edge)           3900.00    3900.00
  clock network delay (ideal)         0.00    3900.00
```

```
input external delay                    1.00    3901.00 f
Start (in)                              0.00    3901.00 f
U42/Y (NOR2X2M)                         0.44    3901.44 r
io/start (IO)                           0.00    3901.44 r
io/U98/Y (NOR2X2M)                      0.08    3901.52 f
io/U24/Y (BUFX2M)                       0.49    3902.01 f
io/U96/Y (NOR2BX2M)                     0.42    3902.43 f
io/U99/Y (NAND2X2M)                     0.14    3902.57 r
io/U97/Y (NAND2X2M)                     0.26    3902.83 f
io/U20/Y (AOI21X1M)                     0.51    3903.34 r
io/U3/Y (AND2X2M)                       0.36    3903.71 r
io/U100/Y (AO2B2X2M)                    0.24    3903.95 r
io/zerocounter_reg[9]/D (DFFRQX2M)      0.00    3903.95 r
data arrival time                               3903.95

clock Dclk (rise edge)                  3906.00  3906.00
clock network delay (ideal)             0.00    3906.00
io/zerocounter_reg[9]/CK (DFFRQX2M)     0.00    3906.00 r
library setup time                      -0.25   3905.75
data required time                              3905.75
-----------------------------------------------------------
data required time                              3905.75
data arrival time                               -3903.95
-----------------------------------------------------------
slack (MET)                                     1.80


Startpoint: firL/SRjm1/out_reg[2]
          (rising edge-triggered flip-flop clocked by Sclk')
Endpoint: firL/outdff/out_reg[39]
         (rising edge-triggered flip-flop clocked by Sclk)
Path Group: Sclk
Path Type: max

Point                                   Incr       Path
-----------------------------------------------------------------------------
clock Sclk' (rise edge)                 15.00      15.00
clock network delay (ideal)             0.00       15.00
firL/SRjm1/out_reg[2]/CK (DFFRHQX1M)    0.00       15.00 r
firL/SRjm1/out_reg[2]/Q (DFFRHQX1M)     0.44       15.44 f
firL/SRjm1/out[2] (Counter10_e_0)       0.00       15.44 f
firL/U19132/Y (AND2X1M)                 0.29       15.73 f
firL/U19134/Y (AND2X1M)                 0.23       15.96 f
firL/U49/Y (BUFX2M)                     0.19       16.15 f
firL/U6597/Y (BUFX2M)                   0.18       16.32 f
firL/U5179/Y (BUFX2M)                   0.25       16.57 f
firL/U1481/Y (AOI22XLM)                 0.27       16.85 r
firL/U444/Y (NAND4XLM)                  0.22       17.06 f
firL/U551/Y (OAI21XLM)                  0.30       17.36 r
firL/U550/Y (NAND4X2M)                  0.18       17.54 f
firL/U20062/Y (NOR4X1M)                 0.26       17.80 r
firL/U90/Y (OA22X1M)                    0.33       18.13 r
```

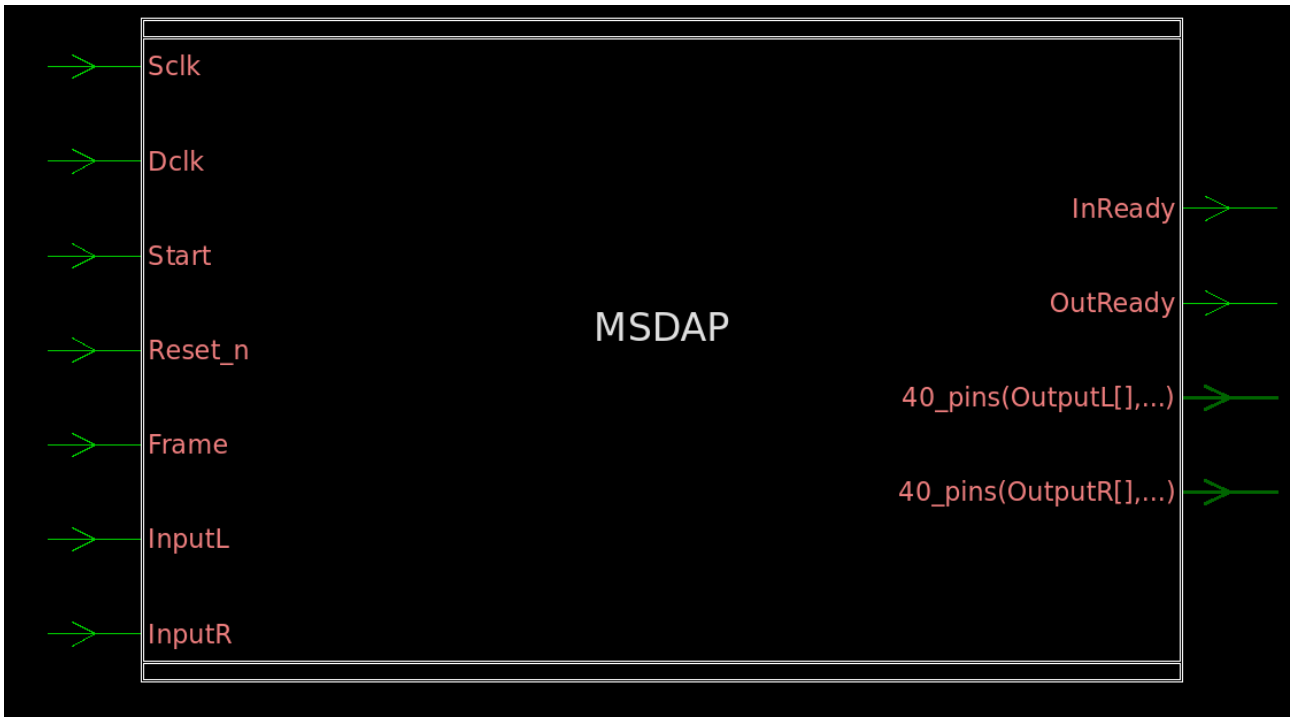
firL/U17132/Y (NOR2X1M) 0.12 18.25 f
firL/U17143/Y (AND2X1M) 0.21 18.46 f
firL/U98/Y (BUFX2M) 0.18 18.64 f
firL/U5211/Y (BUFX2M) 0.17 18.81 f
firL/U112/Y (BUFX2M) 0.16 18.98 f
firL/U122/Y (BUFX2M) 0.27 19.24 f
firL/U149/Y (NAND2XLM) 0.12 19.37 r
firL/U151/Y (AND2X2M) 0.18 19.54 r
firL/U1245/Y (NAND4X2M) 0.15 19.69 f
firL/U17197/Y (OAI21X1M) 0.18 19.86 r
firL/U17198/Y (NAND4X1M) 0.24 20.10 f
firL/U3565/Y (OR4X2M) 0.49 20.59 f
firL/muxxoutxor/two[0] (XOR_1_16_0) 0.00 20.59 f
firL/muxxoutxor/U8/Y (CLKNAND2X2M) 0.08 20.67 r
firL/muxxoutxor/U9/Y (NAND2X2M) 0.07 20.74 f
firL/muxxoutxor/out[0] (XOR_1_16_0) 0.00 20.74 f
firL/mux2/one[0] (Mux4_24_0) 0.00 20.74 f
firL/mux2/U1/Y (AOI22X1M) 0.21 20.95 r
firL/mux2/U16/Y (NAND2X2M) 0.10 21.05 f
firL/mux2/out[0] (Mux4_24_0) 0.00 21.05 f
firL/add/b[0] (Adder24_0) 0.00 21.05 f
firL/add/add_1_root_add_683_2/B[0] (Adder24_0_DW01_add_0)
0.00 21.05 f
firL/add/add_1_root_add_683_2/U1_0/CO (ADDFX2M) 0.59 21.65 f
firL/add/add_1_root_add_683_2/U1_1/CO (ADDFHX1M) 0.29 21.94 f
firL/add/add_1_root_add_683_2/U1_2/CO (ADDFHX1M) 0.28 22.22 f
firL/add/add_1_root_add_683_2/U1_3/CO (ADDFHX1M) 0.28 22.50 f
firL/add/add_1_root_add_683_2/U1_4/CO (ADDFHX1M) 0.29 22.79 f
firL/add/add_1_root_add_683_2/U1_5/CO (ADDFX2M) 0.39 23.17 f
firL/add/add_1_root_add_683_2/U1_6/CO (ADDFHX1M) 0.30 23.48 f
firL/add/add_1_root_add_683_2/U1_7/CO (ADDFX2M) 0.39 23.86 f
firL/add/add_1_root_add_683_2/U1_8/CO (ADDFHX1M) 0.30 24.16 f
firL/add/add_1_root_add_683_2/U1_9/CO (ADDFX2M) 0.39 24.55 f
firL/add/add_1_root_add_683_2/U1_10/CO (ADDFHX1M) 0.30 24.85 f
firL/add/add_1_root_add_683_2/U1_11/CO (ADDFX2M) 0.39 25.23 f
firL/add/add_1_root_add_683_2/U1_12/CO (ADDFHX1M) 0.30 25.53 f
firL/add/add_1_root_add_683_2/U1_13/CO (ADDFX2M) 0.39 25.92 f
firL/add/add_1_root_add_683_2/U1_14/CO (ADDFHX1M) 0.30 26.22 f
firL/add/add_1_root_add_683_2/U1_15/CO (ADDFX2M) 0.39 26.61 f
firL/add/add_1_root_add_683_2/U1_16/CO (ADDFHX1M) 0.30 26.91 f
firL/add/add_1_root_add_683_2/U1_17/CO (ADDFX2M) 0.39 27.30 f
firL/add/add_1_root_add_683_2/U1_18/CO (ADDFX1M) 0.38 27.68 f
firL/add/add_1_root_add_683_2/U1_19/CO (ADDFHX1M) 0.30 27.98 f
firL/add/add_1_root_add_683_2/U1_20/CO (ADDFX1M) 0.37 28.36 f
firL/add/add_1_root_add_683_2/U1_21/CO (ADDFHX1M) 0.29 28.65 f
firL/add/add_1_root_add_683_2/U1_22/CO (ADDFHX1M) 0.29 28.94 f
firL/add/add_1_root_add_683_2/U1_23/S (ADDFX1M) 0.36 29.30 r
firL/add/add_1_root_add_683_2/SUM[23] (Adder24_0_DW01_add_0)
0.00 29.30 r
firL/add/s[23] (Adder24_0) 0.00 29.30 r
firL/mux3/one[39] (Mux2_40_0) 0.00 29.30 r
firL/mux3/U12/Y (AO22X2M) 0.23 29.53 r

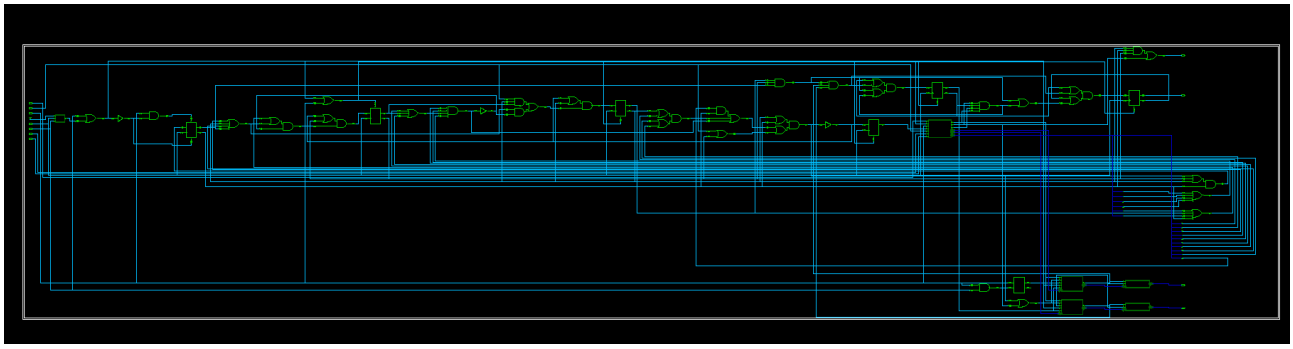
firL/mux3/out[39] (Mux2_40_0)	0.00	29.53 r
firL/outdff/in[39] (DFF40_e_0)	0.00	29.53 r
firL/outdff/U18/Y (AO22X2M)	0.22	29.75 r
firL/outdff/out_reg[39]/D (DFFRQX2M)	0.00	29.75 r
data arrival time	29.75	
clock Sclk (rise edge)	30.00	30.00
clock network delay (ideal)	0.00	30.00
firL/outdff/out_reg[39]/CK (DFFRQX2M)	0.00	30.00 r
library setup time	-0.25	29.75
data required time	29.75	
---		
data required time	29.75	
data arrival time	-29.75	
---		
slack (MET)	0.00	

1

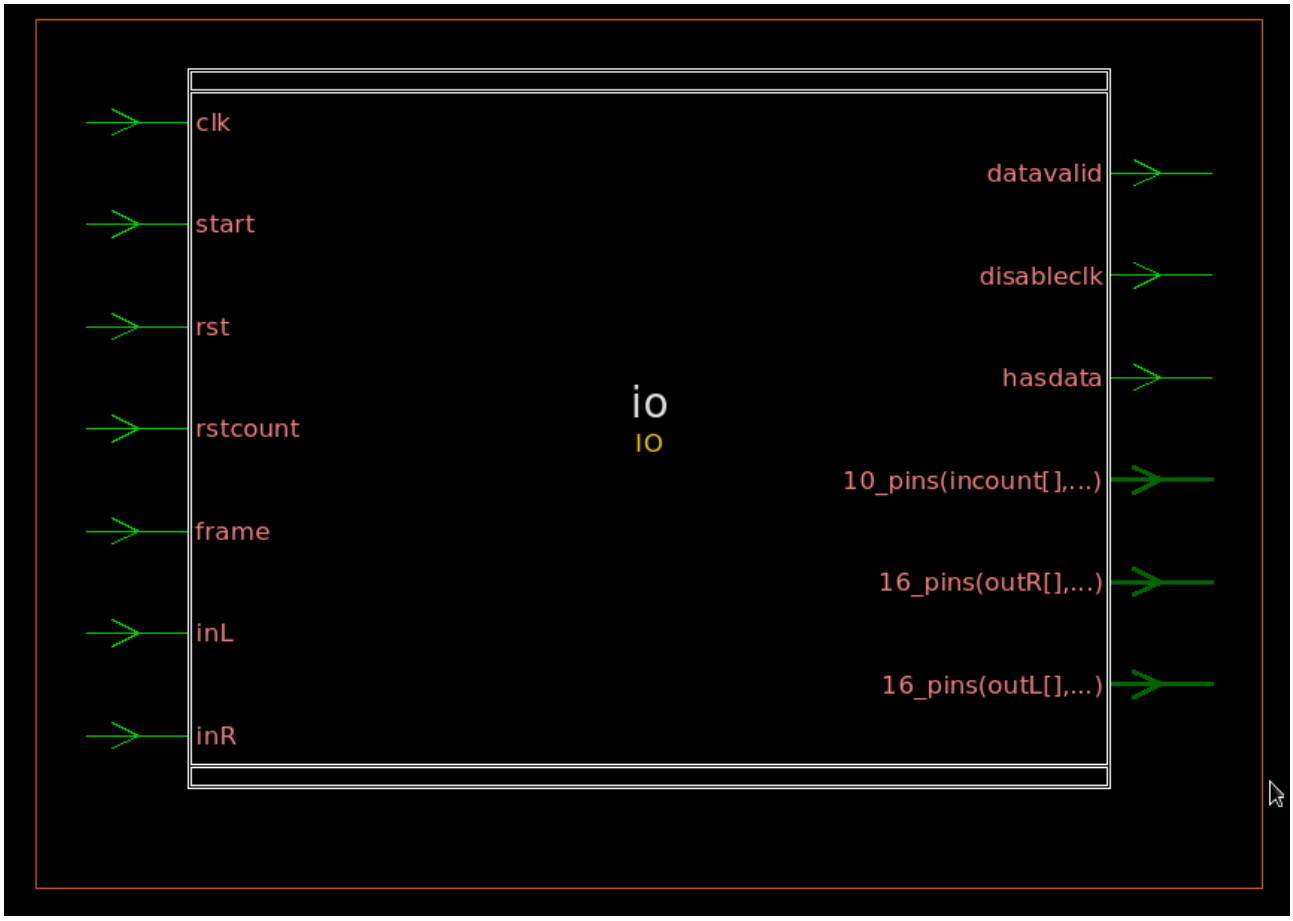
Schematics

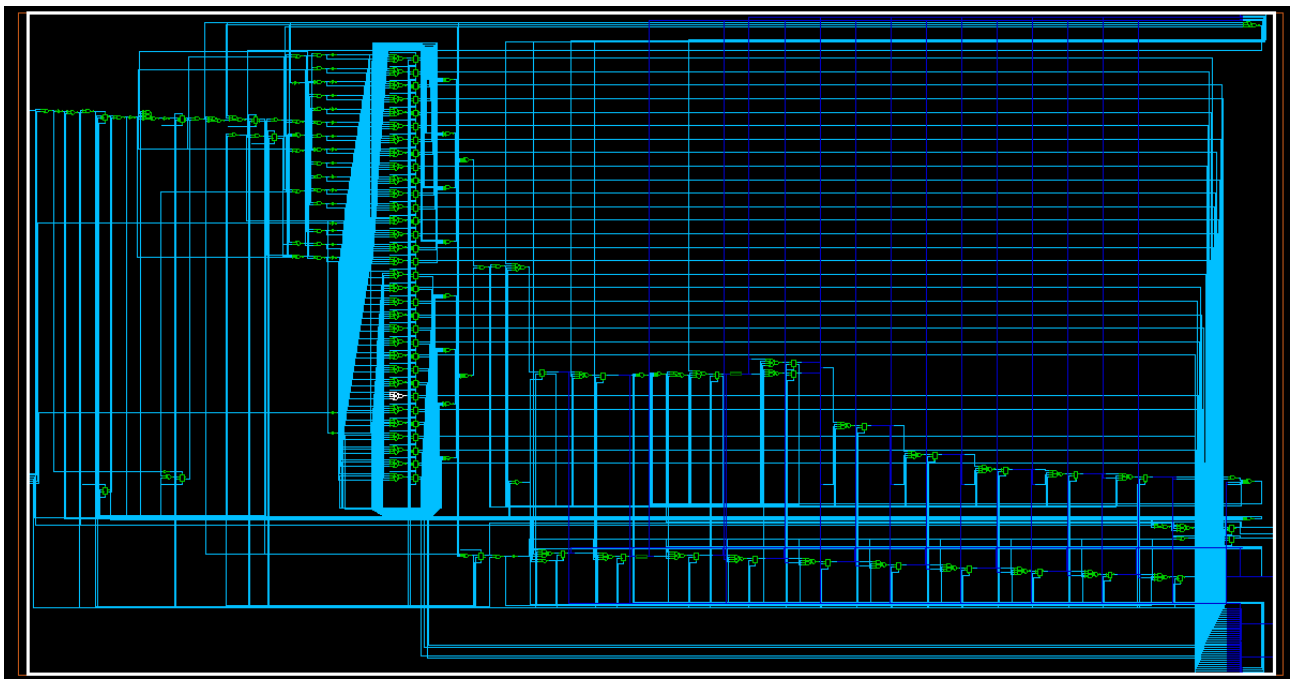
MSDAP





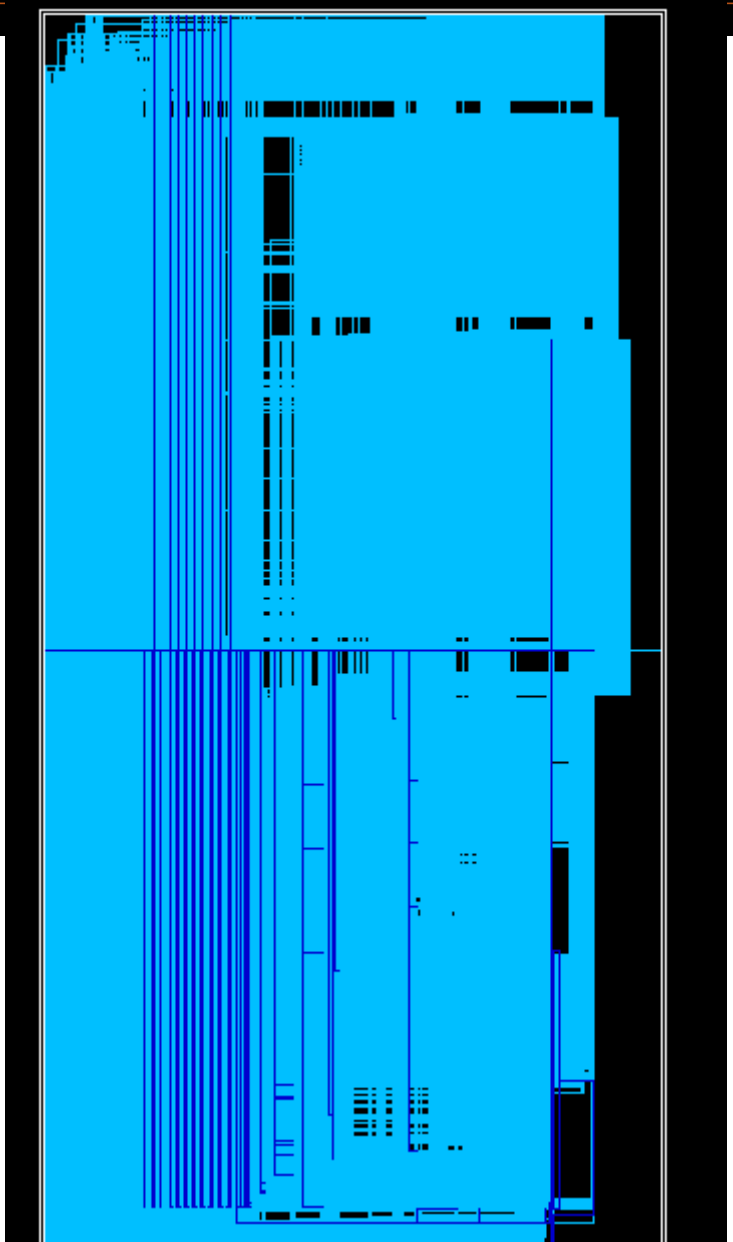
IO


clk
start
rst
rstcount
frame
inL
inR
io
IO
datavalid
disableclk
hasdata
10_pins(incount[],...)
16_pins(outR[],...)
16_pins(outL[],...)

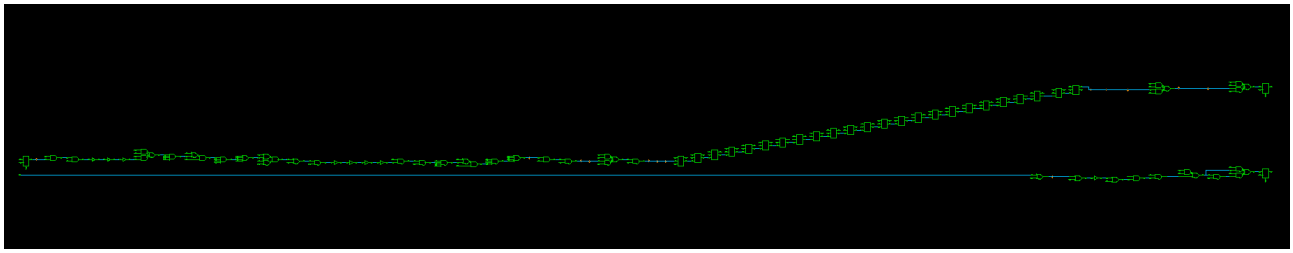


FIR


datain
clk
rst
start
comp
16_pins(x[],...)
firR
FIR_1
dataout
40_pins(out[],...)

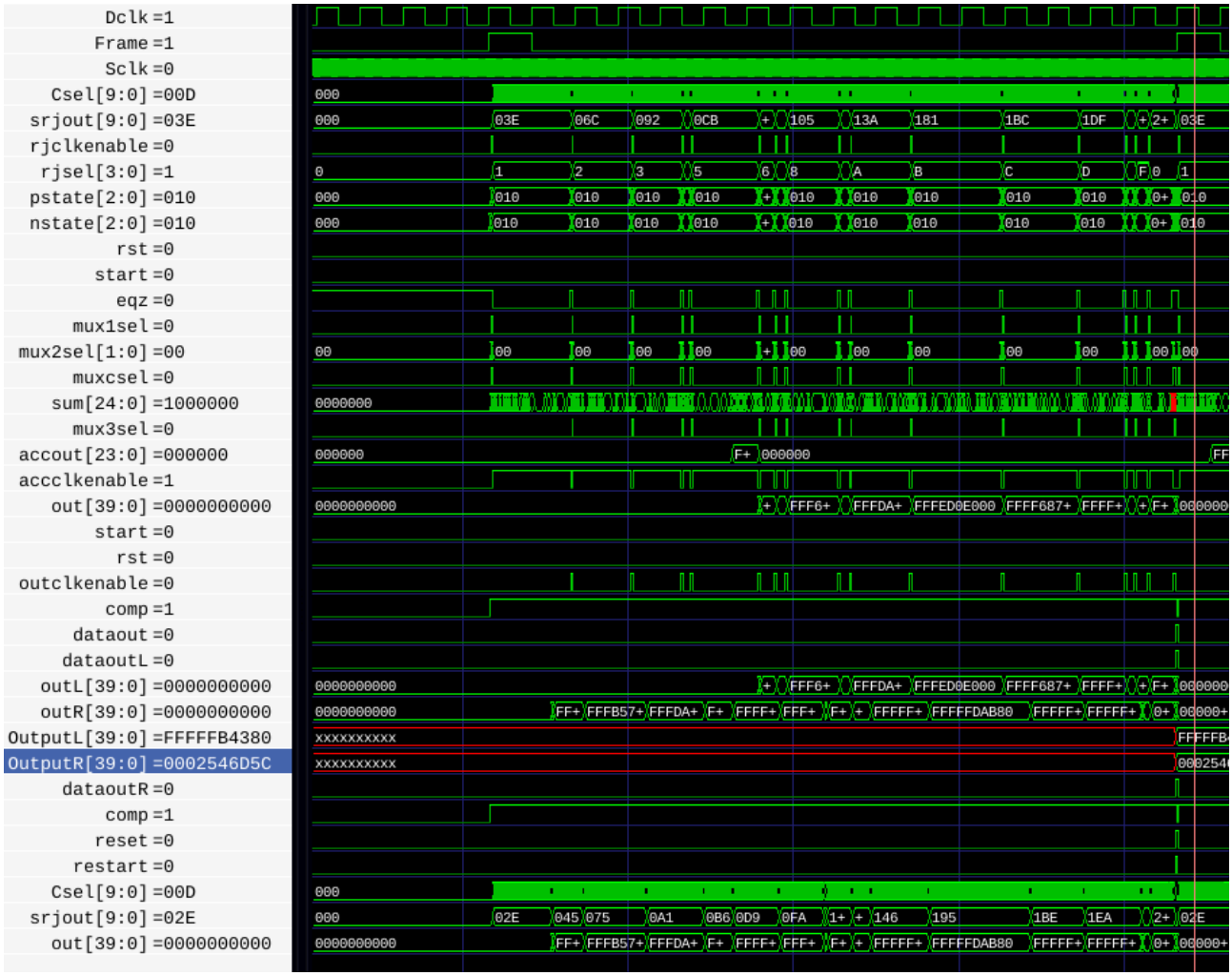


Critical Path

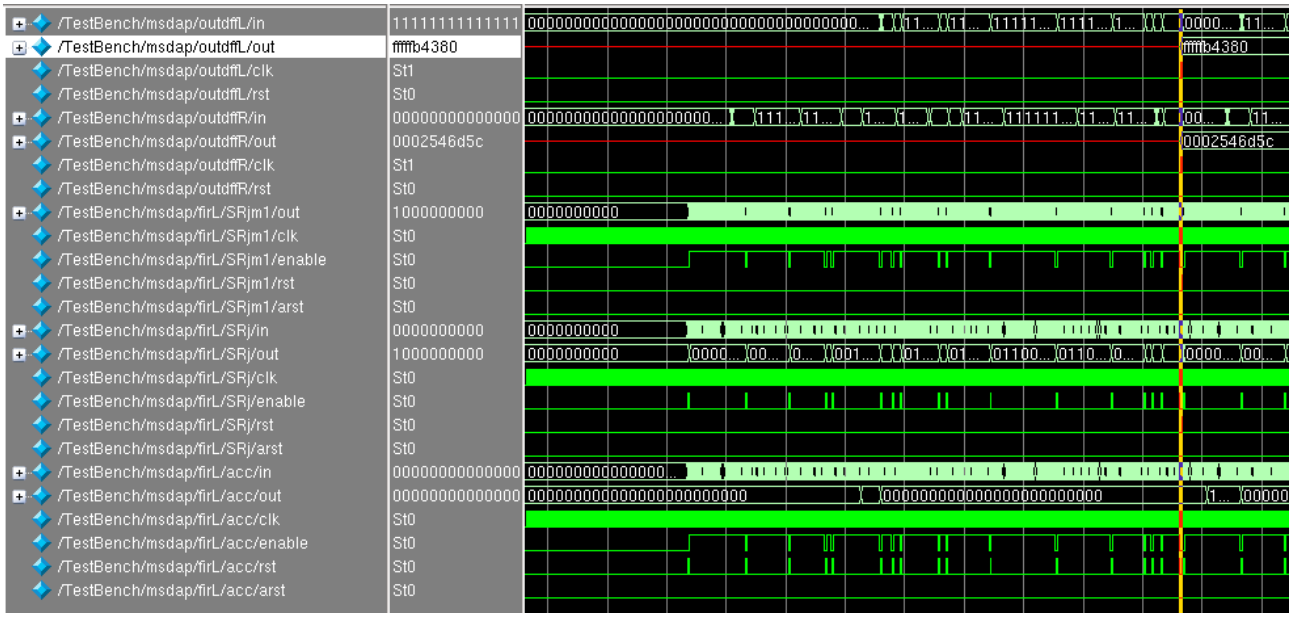


Waves

RTL



Netlist


/TestBench/msdap/outdffL/in
11111111111111
/TestBench/msdap/outdffL/out
ffffb4380
/TestBench/msdap/outdffL/clk
St1
/TestBench/msdap/outdffL/rst
St0
/TestBench/msdap/outdffR/in
00000000000000
/TestBench/msdap/outdffR/out
0002546d5c
/TestBench/msdap/outdffR/clk
St1
/TestBench/msdap/outdffR/rst
St0
/TestBench/msdap/firL/SRjm1/out
1000000000
/TestBench/msdap/firL/SRjm1/clk
St0
/TestBench/msdap/firL/SRjm1/enable
St0
/TestBench/msdap/firL/SRjm1/rst
St0
/TestBench/msdap/firL/SRjm1/arst
St0
/TestBench/msdap/firL/SRj/in
0000000000
/TestBench/msdap/firL/SRj/out
1000000000
/TestBench/msdap/firL/SRj/clk
St0
/TestBench/msdap/firL/SRj/enable
St0
/TestBench/msdap/firL/SRj/rst
St0
/TestBench/msdap/firL/SRj/arst
St0
/TestBench/msdap/firL/acc/in
00000000000000
/TestBench/msdap/firL/acc/out
00000000000000
/TestBench/msdap/firL/acc/clk
St0
/TestBench/msdap/firL/acc/enable
St0
/TestBench/msdap/firL/acc/rst
St0
/TestBench/msdap/firL/acc/arst
St0

Report

Objective

The objective of this project was to synthesize an integrated circuit that implements the MSDAP algorithm. The chip will have 2 clocks: Dclk and Sclk. Dclk must be set at 768 KHz and Sclk may be 26.68 MHz or greater. In order to achieve a clock speed of 26.68 MHz, the algorithm must execute in 560 Sclk clock cycles or less.

Implementation

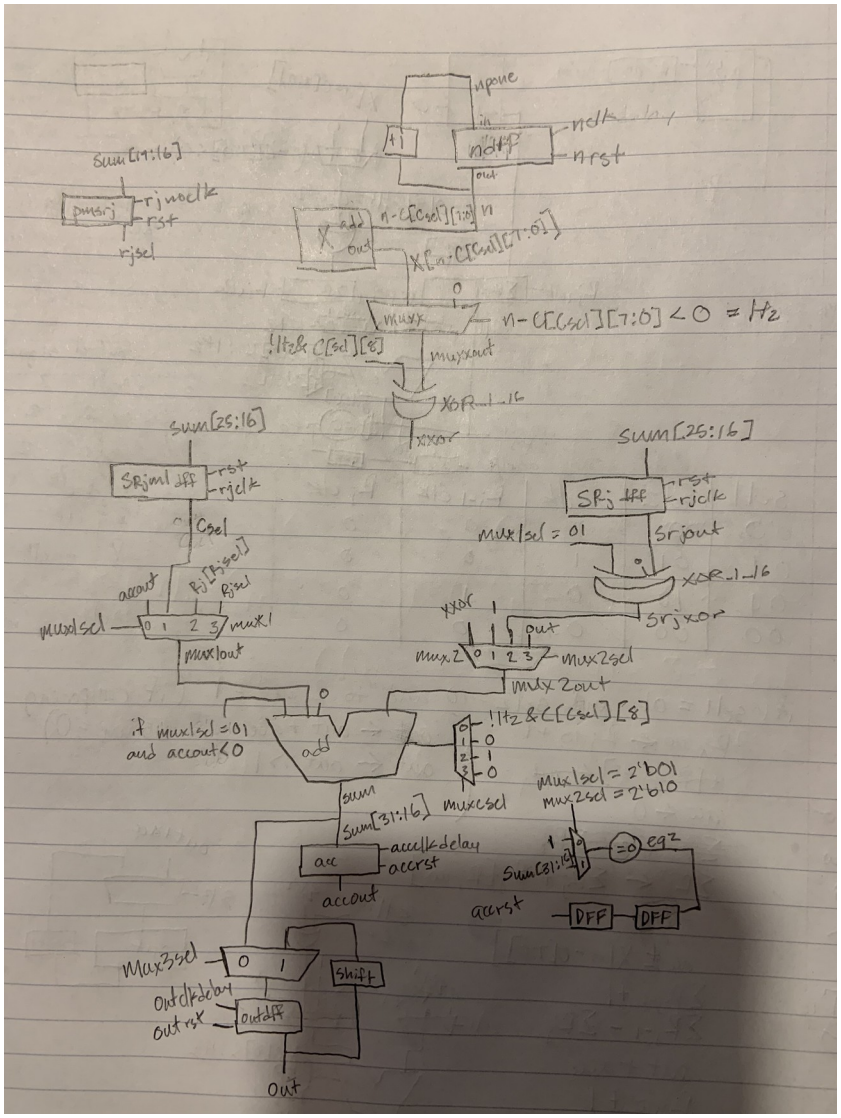
To simulate a clock speed of 768 Khz, a clock with a period of 1302 nanoseconds was used. To simulate a clock speed of 26.68 MHz, a clock with a period of 38 nanoseconds was used. This allows for slightly less than 560 Sclk cycles, but is still possible. Verilog was chosen as the hardware description language and Synopsys' Design Vision was used to compile the design.

$$\left|\frac{1}{768000}\right| = 1.30208333 \times 10^{-6}$$

$$\frac{1}{26680000} = 3.74812594 \times 10^{-8}$$

Design

One quirk of the design that was synthesized is that it will not work with R_j values of zero. This simplifies the design a little, and from what I can tell there isn't a very good reason to have an R_j of 0. Otherwise, I believe that the design correctly implements the MSDAP algorithm.



Results

Given a file 7000 input values, the compiled netlist correctly computed all of them. The input file also contained some resets and some edge cases. So, from my testing, the MSDAP design appears to function properly.

Netlist values compared to expected values.

```
00318d9f23 007ff01b98 | //   5437          00318D9F23      007FF01B98
ffeb0e2223 002540decf | //   5438          FFEB0E2223      002540DECF
001df803d2 000356f3f7 | //   5439          001DF803D2      000356F3F7
ffad530583 ffc15a433e | //   5440          FFAD530583      FFC15A433E
ffdc012903 000db34f04 | //   5441          FFDC012903      000DB34F04
00006c8600 002a29678e | //   5442          00006C8600      002A29678E
000f8c4dd8 ff7b2c1ee8 | //   5443          000F8C4DD8      FF7B2C1EE8
003a07fe90 00659c8d0e | //   5444          003A07FE90      00659C8D0E
ffeadc62be 004032c4ba | //   5445          FFEADC62BE      004032C4BA
```

Special Topic

Developing a Behavioral Model

The first thing to ask when creating a model is “what does the model need to do?” A very obvious question to ask, but necessary. What the model needs to do informs entirely what hardware is needed.

In the case of this project, the MSDAP algorithm needed to be implemented. In the MSDAP algorithm, data is first taken in (R_j, coefficients) and then data is taken in and processed using the R_j’s and coefficients. So registers are needed to store these values. The MSDAP algorithm includes addition and shifting, so of course, an adder and shifter is needed. Any intermediate values between the adds and shifts must be held in registers as well.

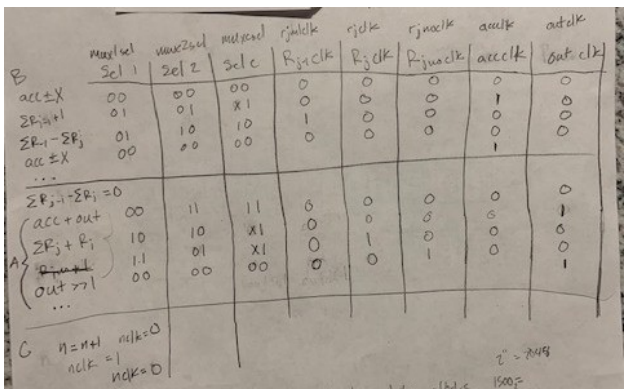
Once a list of all the hardware that is needed is gathered, look at what is required by the hardware. How many actions can be performed in a clock cycle? How many clock cycles do you have to perform the computation? This informs how many functional units your design can or should have. If 15 additions need to be computed in 3 clock cycles, then 5 adders are needed.

Part of the MSDAP project specification was that only one add was allowed per clock cycle, so this means that two adders would waste space since they couldn’t be used in parallel. A soft requirement was that the computation finish in 560 clock cycles. I was initially using the adder to increment a value by 1 and add 2 variables together. The issue was that both these actions needed to be performed 512 times, so a minimum of 1024 cycles was needed to complete the computation. Eventually, I made the increment action executed by a counter. This cut down the minimum amount of time to do these actions to 512 clock cycles.

Now a less foggy picture of the design that needs to be implemented should be made. Which actions need to be performed on which clock cycle? Either write some pseudo code or a state transition diagram. This will inform where multiplexers and flip flops need to be put.

Once the multiplexers and flip flops are placed, this will inform what signals are needed for computation. Look back to the pseudo code to see what actions are needed each cycle and what signals are required for each action. This will inform how the finite state machine is designed.

Shown here are my notes for my first implementation of the finite state machine in the FIR module. On the left is pseudo code, and on the right are the signals required to perform those actions. This helped greatly in designing my finite state machine and I used this chart all the way until the end.



Once all of this is done, your design will not work, because it has a lot of bugs in it. Debugging takes up most of the design process, so have fun doing that.