Lam King Sum Sam (20423789)
Tse Ming Fung Alfred (20756435)

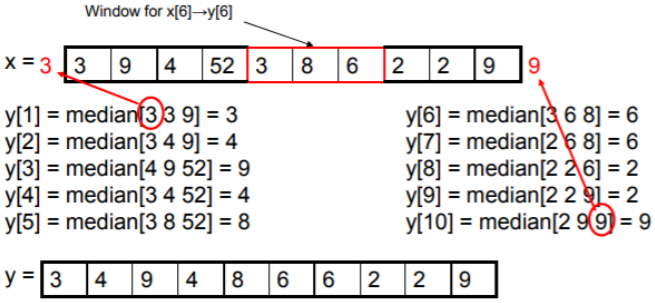
VLSI System Design and Design Automation

Course Project – VLSI Implementation of 1-D median filter

1. Summary

A chip that functions as an efficient 1-D median filter is implemented in this project. The ASIC design flow, from high level specification definition down to final layout extraction and simulation are completed. The area, latency and power under different design parameters are evaluated and discussed too.

A 1-D median filter works with a defined odd window size of N. Such window is sliced along a known signal sequence, and the center value of each window is replaced by the median of the values in the window. The following example shows the application of 1-D median filtering of the signal sequence x[1] to x[10], with N=3 and y[1] to y[10] as output sequence:



A reference architecture is presented in paper [1], to achieve an efficient sorting algorithm to compute a median when the window slices through certain values along the sequence with a throughput of 2, and with area linearly proportional to the window size of N. The Open-Source standard cell library, called Nangate Free PDK 45nm, is used for gate-level netlist synthesis. The designs of different sample resolution {8-bit, 12-bit} and different window sizes {5, 11, 21} are synthesized for comparison.

2. Architecture

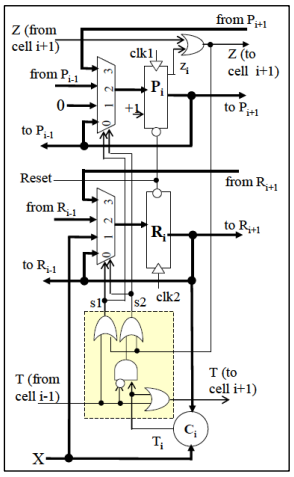


Figure 1 Single cell implementation

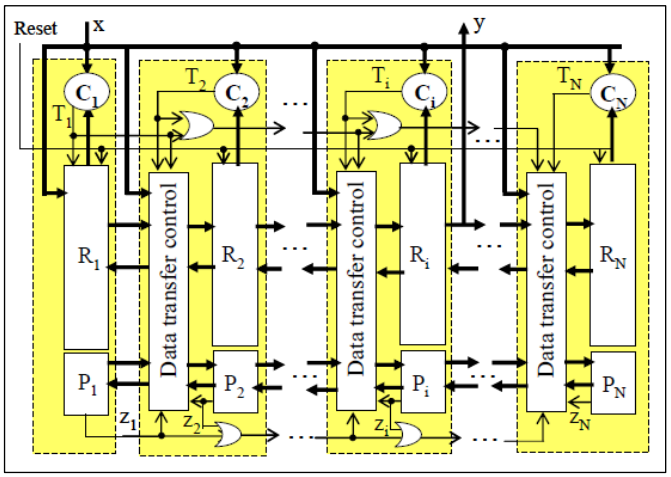


Figure 2 Filter architecture

N cells (each shown in Fig. 1) are cascaded, with each registering one ranked value in the window. The full filter architecture (Fig. 2) stores the samples in descending order, such that the

maximum sample is always at the left (R_1), and the minimum sample at the right (R_N), and the median value (Y) at the center cell ($R_{(N+1)/2}$).

Initially, all registers in the cells are null after a synchronous reset. When a new sample X enters, it is broadcasted to all the cells and compared in parallel with the window samples stored in its own register R. If the incoming sample is larger than the value of register, a signal T is generated to enforce each cell to the right of itself to receive value from its left neighbor, and that cell itself to receive the incoming sample X. Furthermore, a counter P that associate with the register R in each cell keeps track of the "age" of the corresponding value. The reason of keeping track of the age of a value, is that the window should (1) reset the corresponding age value back to zero when X is loaded to R; (2) identify the oldest ranked datum that was swept and discard it to make room for ranking the next new sample. The discard is done by generating a signal Z from the cell that P is incremented up to N, which indicates that the value stored in that cell becomes the oldest data in the window. Z is used to generate logic to enforce each cell to the left of itself to receive value from its right neighbor.

Note that every new sample input X requires only 2 clock cycles to compute the result Y. In the first clock cycle, it increments the counters (P) and removes the oldest datum from the window by moving the stored samples to the left. In the second cycle, it compares the incoming sample to the already ordered window samples and moves some of them to the right to insert the incoming datum in the right place. No matter how large the window is, the circuit preserves the sample ordering obtained at the present cycle, while re-sorting only the new incoming sample at a new clock cycle.

3. Simulations

	X1		X2		X3		X4		X5		X6		X7		X8	
	255		200		10		166		131		59		4		59	
R1	0	255	255	255	255	255	255	255	255	255	200	200	166	166	166	166
R2	0	0	0	200	200	200	200	200	200	200	166	166	131	131	131	131
R3	0	0	0	0	0	10	10	166	166	166	131	131	59	59	59	59
R4	0	0	0	0	0	0	0	10	10	131	10	59	10	10	4	59
R5	0	0	0	0	0	0	0	0	0	10	0	10	0	4	0	4
P1	0	0	1	1	2	2	3	3	4	4	4	4	3	3	4	4
P2	0	0	0	0	1	1	2	2	3	3	2	2	2	2	3	3
P3	0	0	0	0	0	0	1	0	1	1	1	1	1	1	2	2
P4	0	0	0	0	0	0	0	1	2	0	3	0	4	4	1	0
P5	0	0	0	0	0	0	0	0	0	2	1	3	1	0	1	1
CLK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figure 3 Expected values of the window R and counter P in each clock cycle

In the testbench, synchronous reset (`srst`) is set to 1 and the input stimulus X is set to 0 in order to initialize all the states to 0. After the initialization, in each odd number clock cycle, one data will be inserted into X. In each even number clock cycle, median will be calculated which is shown in the highlighted box. To examine the output is valid or not, comparison can be made between the value in the highlighted box and the pins Y[7:0] of the median filter in every even number clock cycle.

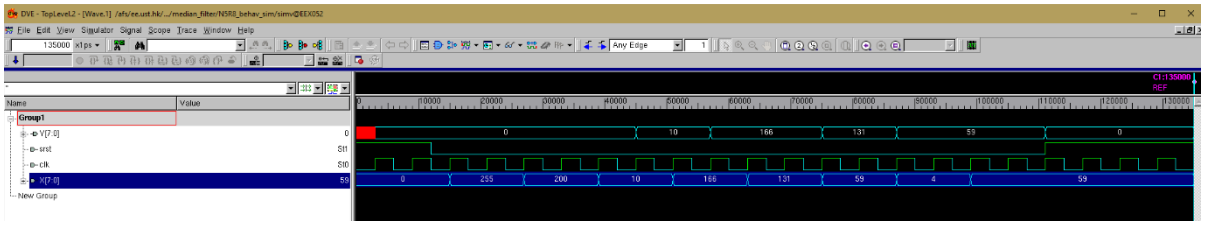


Figure 4 Behavioral simulation for window size=5, sample resolution = 8 bit

Lam King Sum Sam (20423789)
Tse Ming Fung Alfred (20756435)

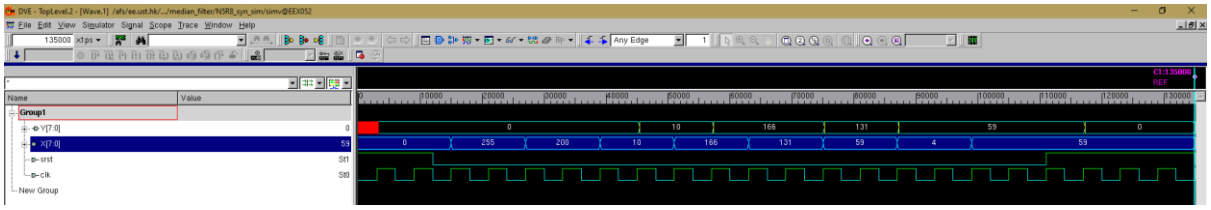


Figure 5 Post synthesis simulation for window size=5, sample resolution = 8 bit

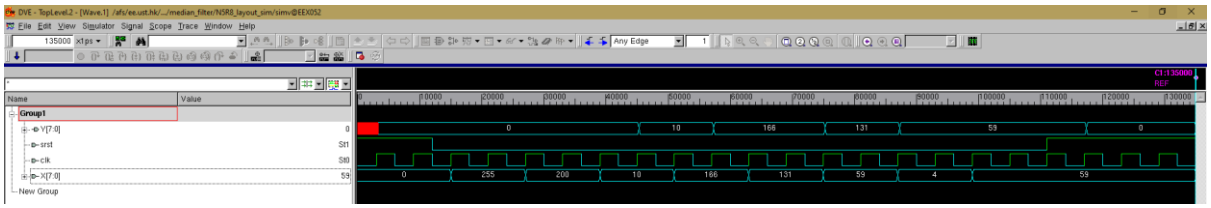


Figure 6 Post layout simulation for window size=5, sample resolution = 8 bit

By comparing the waveforms of behavioral, post synthesis and post layout simulation, all of the results are matched and within our expectation. It shows that the median filter is functioning properly.

4. Post-synthesis results and analysis

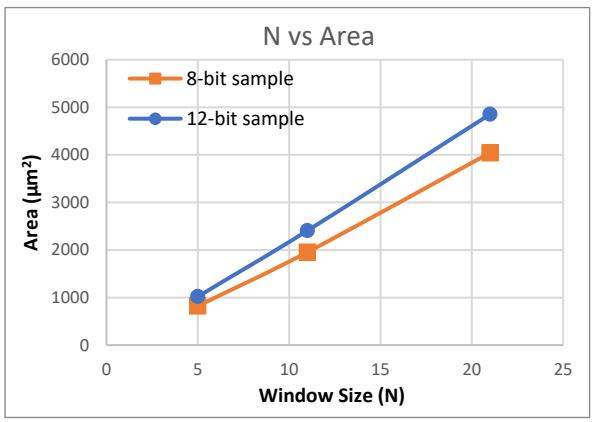
With sample resolution {8-bit, 12-bit} and window sizes {5, 11, 21} vary, six designs are synthesized for comparison. The post-synthesis constraints and results are shown below:

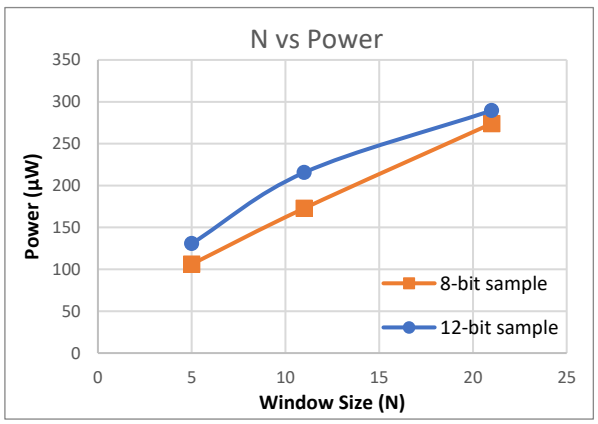
Window size N	Sample Resolution (bit)	Clock period (ns)	Input delay (ns)	Output delay (ns)
5	8	6.0	3.6	3.6
11		8.0	4.8	4.8
21		10.0	6.0	6.0
5	12	6.0	3.6	3.6
11		8.0	4.8	4.8
21		12.0	7.2	7.2

Figure 7 Timing constraints for different window sizes and sample resolutions

Window size N	Sample Resolution (bit)	Area (um^2)	Critical Path (ns)	Maximum clock frequency (MHz)	Power		
					Dynamic (uw)	Leakage (uw)	Total (uw)
5	8	824	2.848	351	188.4276	11.3281	199.7557
11		1946	3.03	330	286.9452	26.8257	313.7709
21		4016	3.84	260	448.6966	54.3332	503.0298
5	12	1024	2.966	337	213.561	13.5206	227.0816
11		2400	3.03	330	341.2324	32.0119	373.2443
21		4854	4.63	216	446.3335	63.555	509.8885

Figure 8 Area, timing and power results for different window sizes and sample resolutions





In each sample resolution, the area and power are plotted against window size N. Area scales up linearly as the window size N increases as expected. The area difference is larger when using 12-bit sample as the window size scales up.

The total power of 8-bit sample designs has linear relationship with the window size, but for 12-bit sample, N=21 design consumes less power than expected. This may be due to that synthesis is constrained with relaxed clock period of 12ns, so the critical path in this design is longer than expected, thus a lower maximum clock frequency, and a lower dynamic power estimation. Also doubling the area does not double the power consumption the design, which the performance (window size means the ability to sort N samples at the same time) may be worth of a trade-off with power consumption.

5. Post-layout result

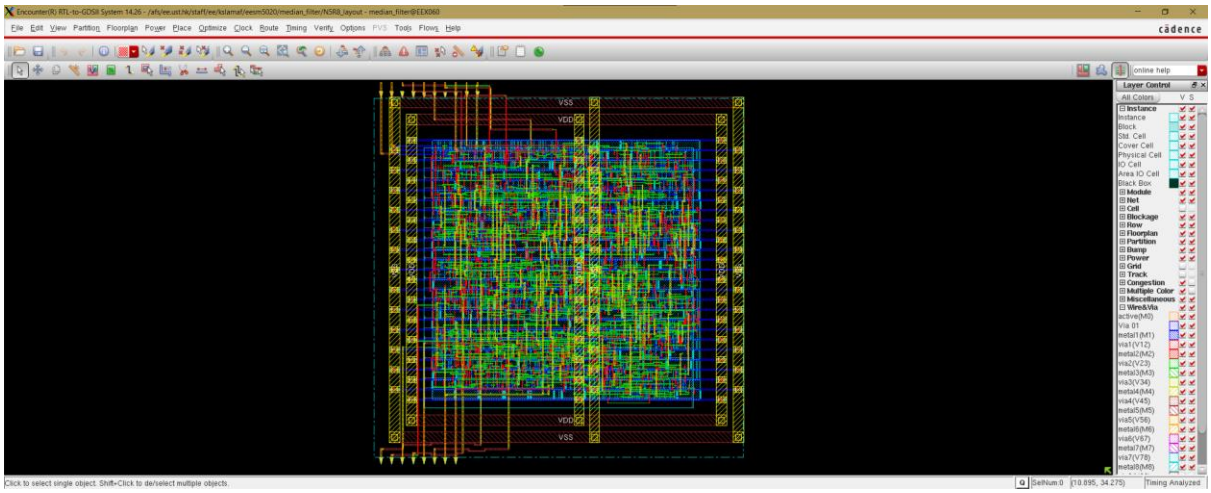


Figure 9 Layout for window size=5, sample resolution = 8 bit

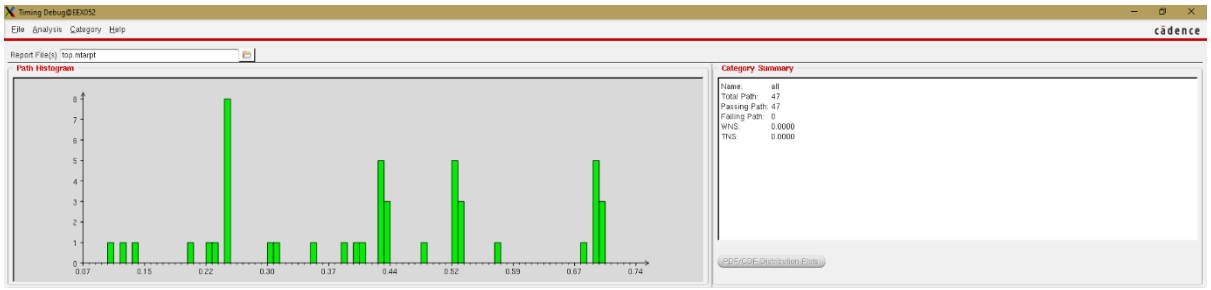


Figure 10 Setup time path histogram

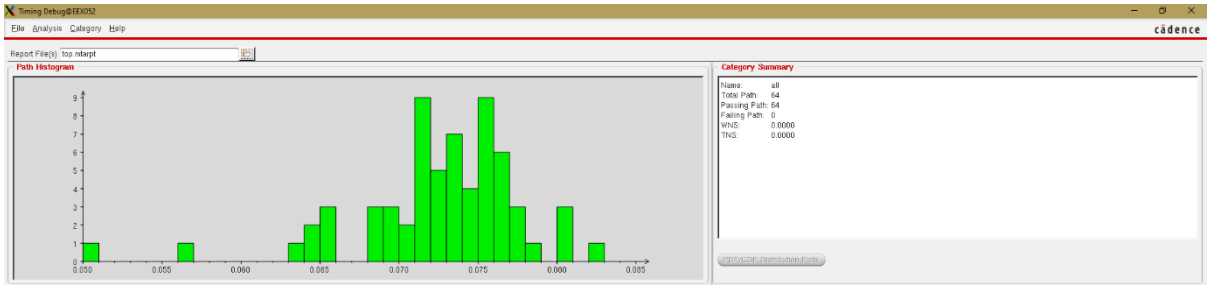


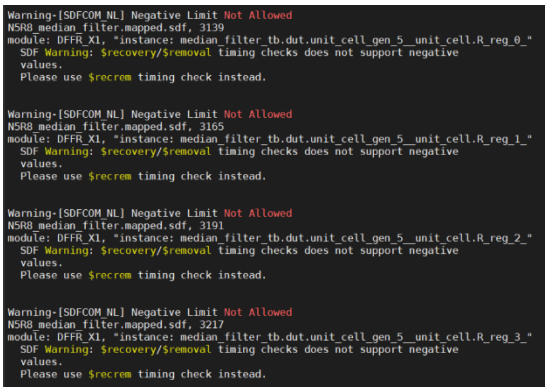
Figure 11 Hold time path histogram

6. Challenges

The RTL is coded in a way that the window size (N) can be easily parameterized. `unit_cell`

modules are instantiated by using the `generate` reserved word from Verilog. Since some of the port connections of the first and last cell are different from the middle cell(s), the same cell was instantiated with some ports tied to 0 or left un-connected. This causes some synthesis warnings but should not affect the computation (proven by post-layout simulation).

During synthesis, we found warnings on the DFFR_X1 cell from the Free PDK 45nm and later find out that that the D-flipflops in such library does not support asynchronous reset:



All the resetting mechanism of the registers in the RTL need to be replaced with synchronous reset before re-doing synthesis. This also affect when to drive the reset signal in the testbench.

7. Conclusion

A 1-D median filter (N=5 and sample width of 8-bit) has been implemented as a chip. The layout design has been verified under testbench. With 45nm process, the area is 827.792µm2, can run up to 351MHz and consuming 199.8µW. Since the sampling can update every other clock cycle, the maximum sampling rate is 175.5MHz. Increases the window length of the filter linearly scales up the area.

Reference

[1] Vasily G. Moshnyaga and Koji Hashimoto, “An Efficient Implementation of 1-D Median Filter”, in the Proceedings of 2009 IEEE International Midwest Symposium on Circuits and Systems