

The Application of a Hierarchical Top-Down Parameterized VHDL Design Methodology to a Carrier Recovery Loop Filter ASIC

Lori Lee, Kim Guzzino, and Alan Kwentus

Ground Systems Center, TRW
One Space Park R6-2583
Redondo Beach, CA 90278
(310)814-1055
email:llee@emu.sp.trw.com

Abstract

This paper discusses a hierarchical top-down parameterized VHDL design methodology [1] and its application to the design of a digital carrier recovery loop filter. The loop filter ASIC includes all critical functions such as multipliers, delay registers, accumulators, counters and adders, all of which were synthesized from a parameterized VHDL description. In the process of a top-down system design approach, design trade-offs are usually difficult due to the lack of information about the detailed hardware implementation. This paper shows through an example how hardware specifications such as gate count and critical path were obtained quickly to assist in the system level trade-offs. An analysis of area vs. speed has helped to identify the most optimal parameters which will yield the most efficient design.

1.0 Introduction

The current carrier recovery loop circuit in a demodulator/bit synchronizer developed at TRW is implemented with off-the-shelf components, operating at low speed with high power dissipation. By implementing the carrier recovery loop filter and the

phase/lock detector in an ASIC, the carrier recovery loop processing speed can be increased by a factor of 10, the size of the carrier recovery loop circuit board can be reduced by half, and reliability can be improved. A hierarchical top-down parameterized VHDL design methodology was utilized to implement such an ASIC. Parameterized VHDL allows inexpensive future technology upgrades and is reusable for different system applications. Hierarchical VHDL increases the readability and feasibility of the code, eases the debugging process, and shortens the simulation time.

2.0 Approach

The growing time-to-market pressure has helped to promote the popularity of top-down design methodologies and logic synthesis tools. The top-down design methodology approach is to define the abstract functionality of the circuit at a very high system level rather than at the gate-level. Once the functionality is defined, logic synthesis tools are used to implement the circuit at the gate-level. In the process of defining functionality, it is useful to create hierarchy to structure the VHDL code by partitioning the design according to function. The partitioned design will yield

smaller blocks which are easier to debug, modify, and simulate. It also increases routability when the circuit is ready for the place and route because the structural design can be floorplanned easier than the overall top-level design.

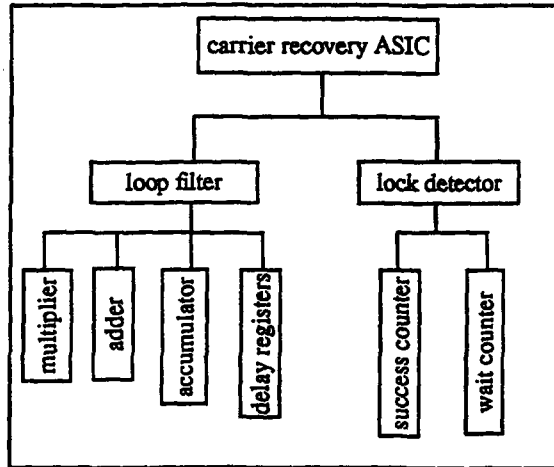


FIGURE 1. Carrier Recovery ASIC Hierarchy

Once the hierarchy is defined, the next step is to identify the components of the design and to create the VHDL models of the components. There are four major components in the loop filter:

- multiplier
- delay registers
- accumulator
- adder

and two major components in the phase/lock detector:

- success counter
- wait counter

Each component's VHDL is coded with parameterized ports for the input, output, and internal bit widths. This is accomplished using the VHDL language *generic* construct. The actual parameters and the vendor library are specified upon synthesis.

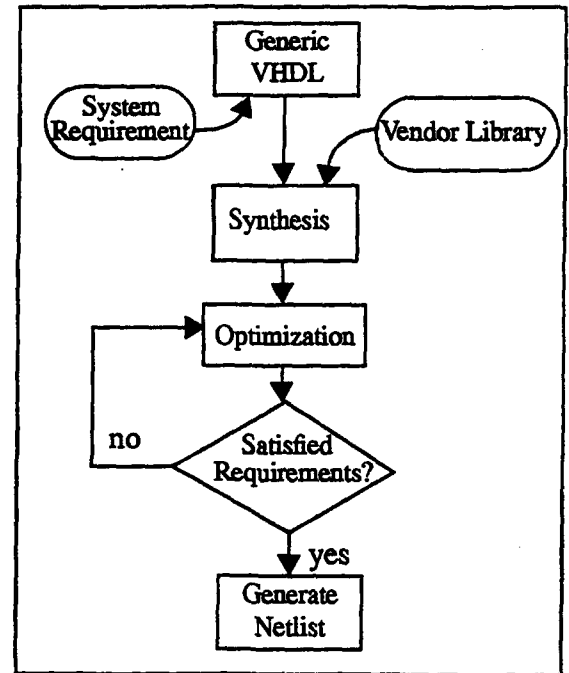


FIGURE 2. VHDL Design Flow

VHDL is accepted as a universal input by various CAD tools including: a.) synthesis tools to synthesize the circuit with any desired vendor library, b.) simulation tools to estimate the gate count, critical path, and power dissipation of the synthesized design, and c.) optimization tools to optimize parameters such as area and speed.

For this application, Synopsys was chosen as the synthesis, optimization, and simulation environment. Vitesse Semiconductor was chosen as the IC vendor. GaAs was chosen as the implementation technology primarily due to the required high-speed of operation. Once the requirements and vendor are specified, the circuit is ready for synthesis. After synthesis, the design is verified and, if necessary, goes through several iterations to optimize the critical design constraints (size, speed, power, etc.).

The work described in this paper did not stop at the point where the design requirements were met. Instead, three parameters (area, speed, and bit width) were varied

over a wide range and the data collected from the design iterations was used to help the system designer make rapid system-level trade-offs with accurate knowledge of the effects on the detailed implementation.

3.0 Design Trade-Offs

3.1 Pipelining

With most state-of-the-art logic synthesis tools, VHDL allows design trade-offs, such as area vs. speed, at a very early stage in the design cycle. In this particular application, the ASIC is required to run at 200 MHz (5 ns critical path) with a latency less than 70 clock cycles. Since it is very difficult to determine the wire load RC delays due to routing prior to performing place and route, the first design iteration (#1) pipelined the circuit at every level of logic in order to achieve the fastest possible speed. The downside to this approach was the significant increase in gate count due to the large number of pipeline registers in the design. The result was an ASIC requiring 47K gates, with a 1.82 ns critical path, and a 78 clock cycle latency. Because this first iteration failed the latency requirement, the second iteration (#2) concentrated on reducing the gate count and latency by pipelining at every two levels of logic. The gate count was reduced to 33K and latency was reduced to 47 clock

cycles, but the critical path was increased to 2.82 ns.

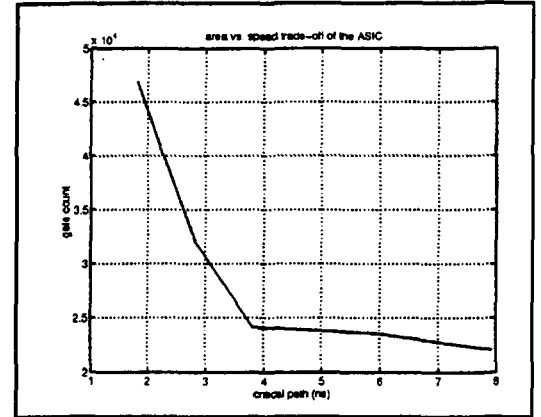


FIGURE 3. ASIC Area vs. Speed at Different Pipelining Stage

Even though iteration #2 met the requirements and was a better design than iteration #1, further design iterations were performed to find the most efficient design. In other words, what is the optimal point where reducing pipelining registers does not yield significant reduction in gate count but only worsens the critical path? The results presented in Table 1 and Figure 3 show that iteration #3 is the optimal point where the design achieves the most efficient gate count and speed.

This gate count vs. speed trade-off is made possible by the VHDL design methodology and the versatility of the logic synthesis tools. By changing the number of appended output registers in the VHDL

TABLE 1. Trade-off of the ASIC

iteration	pipeline stage	latency	gate count	critical path
#1	every logic level	78	46,874	1.82 ns
#2	every 2 logic levels	47	32,116	2.82 ns
#3	every 3 logic levels	36	24,144	3.89 ns
#4	every 4 logic levels	31	23,527	5.92 ns
#5	every 8 logic levels	25	22,062	7.93 ns

code and then commanding the synthesis tool to balance the appended registers into the circuit core, one can easily perform circuit retiming and pipelining to cut down the critical path delay. This technique is widely used in DSP applications.

3.2 Interleaving

Another approach to reduce the gate count is to use an *interleaving* technique for the multipliers. The two multipliers of the ASIC are identical: one for multiplication

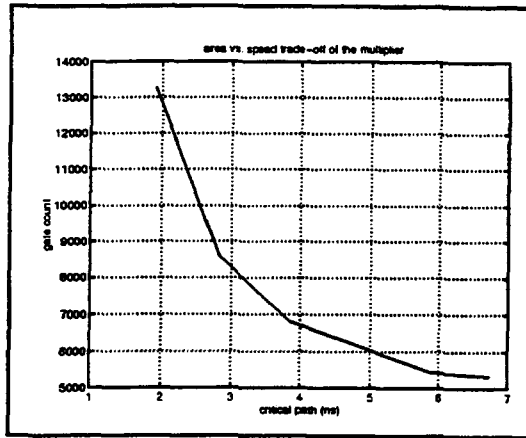


FIGURE 4. Multiplier Area vs. Speed at Different Pipelining Stage

of the proportional constant (K_p), and the other for multiplication of the integral constant (K_i) as shown in Figure 5 (top figure). Interleaving uses one multiplier operating at twice the speed to handle both multiplications, with one of the multiplier's inputs multiplexed between the proportional and integral constant. The high-speed operating clock is used as the select signal of the multiplexer as shown in Figure 5 (bottom figure). Table 2 shows several design iterations for pipelined multipliers. For this application where the circuit is required to operate at a 200 MHz data rate, the design of iteration #1 in Table 2 would be required (1.9 ns critical path). Therefore interleaving does not provide a significant gate count reduction since 2 copies of design #3

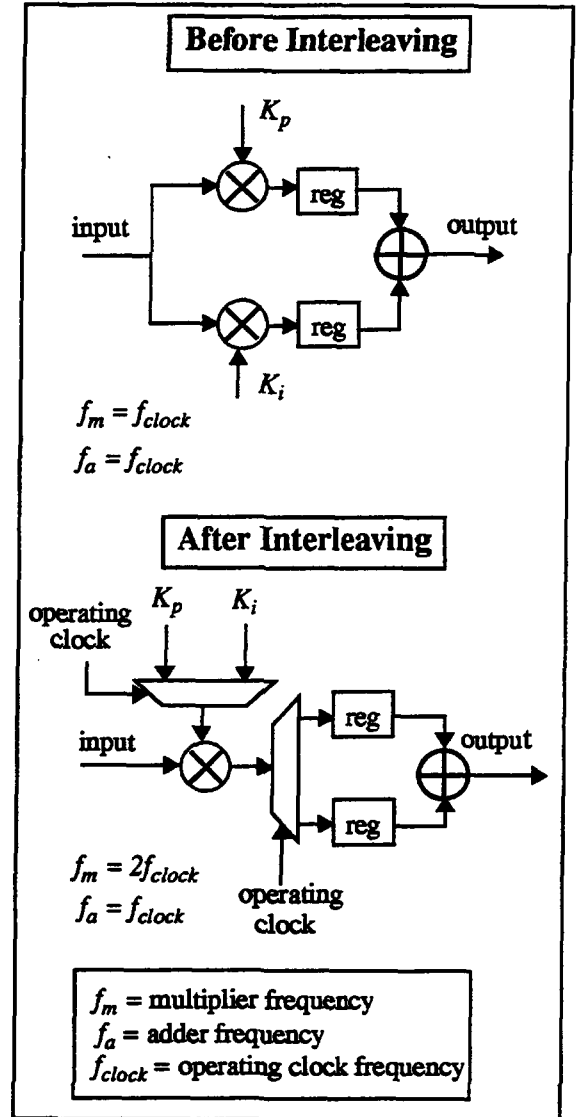


FIGURE 5. Circuit Before & After Applying Interleaving Technique

(6,802 gates, 3.86 ns critical path) requires 13,604 gates whereas one copy of design #1 requires 13,280 gates for comparable overall performance.

If the speed requirement had been 100 MHz instead of 200 MHz, however, this approach would achieve a significant gate reduction of approximately 4,000 gates since the filter can be implemented with a single copy of the #3 iteration multiplier instead of two #5 iteration multipliers ($5,336 \times 2 - 6,802 = 3,870$) (Table 2).

TABLE 2. Trade-off of the multipliers

iteration	pipeline stages	latency	gate count	critical path
#1	every logic level	18	13,280	1.9 ns
#2	every 2 logic levels	10	8,586	2.82 ns
#3	every 3 logic levels	4	6,802	3.86 ns
#4	every 4 logic levels	2	5,436	5.88 ns
#5	every 8 logic levels	1	5,336	6.73 ns

This example shows how the VHDL design methodology helps the designer to analyze the pros and cons of different techniques so that the best architecture for a given application and set of requirements can be realized.

3.3 Bit Width

The ASIC has been designed to accommodate a 12-bit 300 MHz DAC that is currently available. However, when a better DAC becomes available, the ASIC can be upgraded rapidly since the parameterized

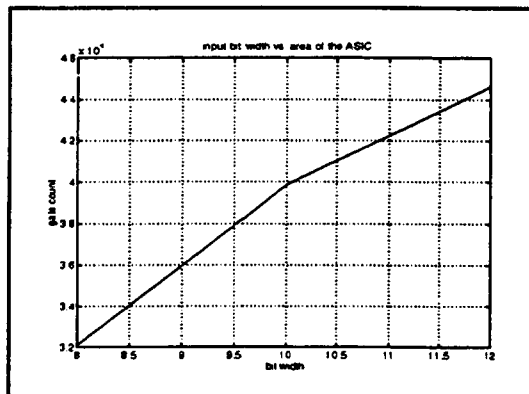


FIGURE 6. ASIC Area with Different Input Bit Width

VHDL design methodology allows the output bit width of the ASIC to be specified at the time the design is synthesized to gates. In the simplest case, only the *generic* constant is adjusted. For a more optimal design where pipelining registers are used, it might also be necessary to modify the number of pipeline registers added at the output of the circuit. Table 3 shows how

the input bit width affects the gate count for this example. Figure 6 shows that the gate count increases almost linearly as the input bit width increases.

TABLE 3. ASIC Area vs. Input Bit Width

input bit width	latency	gate count	critical path
8-bit	47	32,116	2.82 ns
10-bit	48	39,852	2.82 ns
12-bit	49	44,620	2.82 ns

4.0 Frequency Response

The functionality and timing of the ASIC was confirmed through gate-level simulation. However, it is also desirable to determine the performance of the filter through a comparison of the simulated and theoretical step response of the filter. A block dia-

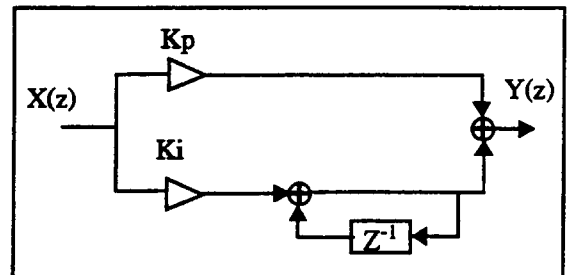


FIGURE 7. Block Diagram of Carrier Recovery Loop Filter

gram representation of the filter is shown in Figure 7. The triangle represents the gain while the rectangle denotes a delay register. The latency of the ASIC does not

affect the frequency response of the loop filter. It will only add delay in the phase response. The only delay register that will affect the frequency response is the accumulator which stores the previous value of the product of the phase error and integral constant.

The transfer function of the implemented loop filter in the Z-domain can be expressed as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + \frac{K_i}{K_p} - Z^{-1}}{1 - Z^{-1}} .$$

The filter has a

zero at $Z = \frac{1}{1 + K_i/K_p}$ and a pole at $Z = 1$.

The frequency response of the transfer function can be obtained by substituting

$Z = e^{j2\pi f}$ in $H(z)$. The theoretical frequency response of the loop filter is shown in Figure 8.

The frequency response of the experimental data can be obtained by taking the Fourier transform of the simulated output of the ASIC. Due to bit truncation after the multipliers, accumulators, and adders (a total of 22 bits), it is impossible to observe any output of the integral branch of the circuit unless the input and the integral constant are at their largest magnitude and are held steady for at least 3 clock cycles. Therefore, a step response input was used to force the output. From Figure 9, it is clear that the loop filter is a low pass filter. The discrepancy of the simulated and the theoretical data was mainly due to the bit truncation.

5.0 Summary

A loop filter is a basic building block used in phase locked loops for carrier recovery in communication systems. A general pur-

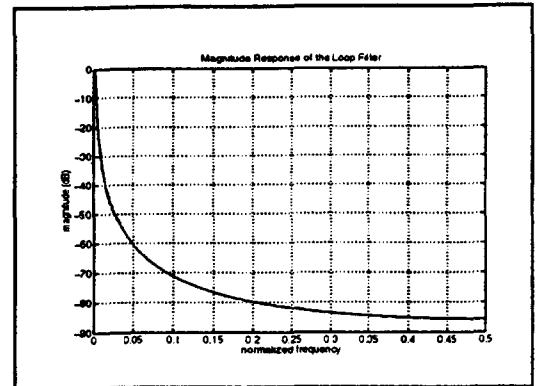


FIGURE 8. Theoretical Frequency Response of Loop Filter

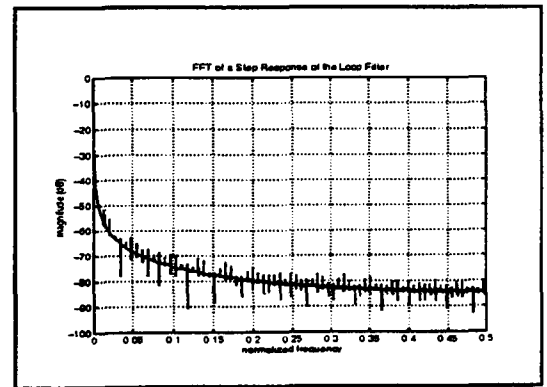


FIGURE 9. Simulated Frequency Response of Loop Filter

pose loop filter block can be created using a hierarchical parameterized VHDL design methodology. That is, the VHDL model is designed with parameters for certain implementation details, (such as bit widths) that are specified at the time the model is synthesized to a gate-level implementation. This allows a single parameterized VHDL model to be used across several applications requiring different parameters. It also facilitates rapid high-level system trade-offs by providing accurate gate count and critical path timing information for various parameter options. Furthermore, the parameterized VHDL model is inherently technology independent because it can be synthesized to an implementation in any available gate-array or standard cell library through the use of CAD hardware synthesis tools (such as Synopsys). These CAD tools also allow

optimization of area, speed, and power dissipation, depending on the specific application's critical design goals. It is the use of these optimizing CAD tools that enables the rapid system design trade-offs as well as pin-pointing the bottlenecks in the design.

References

- [1] A. Kwentus, K. Guzzino, and D. Thornhill, "A parameterized VHDL design methodology and its application to the implementation of high-performance FIR filters", *Proceedings of the 1993 Government Microcircuit Application Conference*, vol. XIX, pp. 177-178, November 1-5, 1993