DLL-Based Frequency Multiplier

Final Project Report
VLSI Chip Design Project

Project Group 4
Version 1.0

Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewed</td>
<td>Ameya Bhide</td>
</tr>
<tr>
<td>Approved</td>
<td>Ameya Bhide</td>
</tr>
</tbody>
</table>
# PROJECT IDENTITY

Project group 4, 2012/Spring  
Linköping University, ISY, Electronic Devices

<table>
<thead>
<tr>
<th>Name</th>
<th>Responsibility</th>
<th>Phone</th>
<th>E-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Henrik Larsson</td>
<td></td>
<td>073-841 30 32</td>
<td><a href="mailto:henla473@student.liu.se">henla473@student.liu.se</a></td>
</tr>
<tr>
<td>Oscar Marklund</td>
<td></td>
<td>070-684 35 62</td>
<td><a href="mailto:oscma009@student.liu.se">oscma009@student.liu.se</a></td>
</tr>
<tr>
<td>David Moberg</td>
<td>Project leader (PL)</td>
<td>070-393 84 49</td>
<td><a href="mailto:davmo579@student.liu.se">davmo579@student.liu.se</a></td>
</tr>
</tbody>
</table>

**Course responsible:** Atila Alvandpour, atila@isy.liu.se  
**Customer and Supervisor:** Ameya Bhide  
Office: B-house 2B:229:218, 581 00 LINKÖPING, Phone: 013-288946, Fax: 013-139282  
Email: ameya@isy.liu.se
Introduction
System Description
  Timing
Block Level and Description
  Phase Detector
  Clock Divider
  Counter
  Delay Line
  Transition Detector
  Edge Combiner
    Symmetric AND
    TPL
Simulation Results
PAD Assignments / Early Test Plan
  Pins
Risks and Delays
Project Evaluation
References
Time Report
Introduction

We are using a pure digital approach to realize a DLL-based frequency multiplier. It uses a multiplier of 8x and is primarily designed for an output frequency of 1GHz, although the actual output range is subject to process variations.

The input frequency range designed for at the start was approximately 125-250 MHz for an 8x multiplier and 250-500 MHz for a 4x multiplier, enabling an output frequency between 1 GHz and 2 GHz. But due to problems with the sizing of the delay line the input range is now around 100-130 MHz with variations depending on transistor corners. There is also no option for a 4x multiplier since it was no longer needed.
System Description

The different subcomponents and their interconnects can be seen in figure 1.

Figure 1. The complete core layout with buffers.
Timing

The minimum cell delay in post-layout is 456 ps and the maximum cell 657 ps, which results in an input frequency range of 95-137 MHz. This is overall lower than the numbers from our transistor-level simulations, since simulations on the layout level showed that the TPL could output a maximum of only slightly above 1 GHz.

<table>
<thead>
<tr>
<th>Cell delay [s]</th>
<th>Multiplier</th>
<th>Last bit</th>
<th>Period [s]</th>
<th>Input frequency [Hz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.56E-10</td>
<td>8x</td>
<td>16</td>
<td>7.30E-09</td>
<td>1.37E+08</td>
</tr>
<tr>
<td>6.57E-10</td>
<td>8x</td>
<td>16</td>
<td>1.05E-08</td>
<td>9.51E+07</td>
</tr>
</tbody>
</table>

Schematic

The minimum cell delay is now 310 ps and the maximum cell delay is 490 ps, which results in an input frequency range of 128-202 MHz. This differs a bit from our initial design goals, since simulations showed that the TPL can output a maximum of 1.5 GHz, and the minimum will be lower once the cells are connected to each other.

<table>
<thead>
<tr>
<th>Cell delay [s]</th>
<th>Multiplier</th>
<th>Last bit</th>
<th>Period [s]</th>
<th>Input frequency [Hz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10E-10</td>
<td>8x</td>
<td>16</td>
<td>4.96E-09</td>
<td>2.02E+08</td>
</tr>
<tr>
<td>4.90E-10</td>
<td>8x</td>
<td>16</td>
<td>7.84E-09</td>
<td>1.28E+08</td>
</tr>
<tr>
<td>3.10E-10</td>
<td>4x</td>
<td>8</td>
<td>2.48E-09</td>
<td>4.03E+08</td>
</tr>
<tr>
<td>4.90E-10</td>
<td>4x</td>
<td>8</td>
<td>3.92E-09</td>
<td>2.55E+08</td>
</tr>
</tbody>
</table>
Block Level and Description

Our standard sizes for transistors was 0.6 µm and 1.55 µm for NMOS and PMOS respectively. When we needed bigger transistors we primarily used multiples of these values. One exception is the TPL which has very custom sized transistors.

Some transistor sizes were tweaked from the schematics since post-layout simulations are generally slower than schematic-simulations.

Figure 2. Block level design of the system.
**Phase Detector**

This block detects if the delayed clock signal is aligned to the input clock. This is achieved with a simple D Flip-Flop clocked on the input clock. If the delayed clock is already 1 at this time it means it should be delayed more, sending out an up signal. Otherwise the delay is too much and should be decreased. The setup time of this DFF limits the resolution of the clocks’ alignment which means that it’s critical to make this DFF fast enough to detect the smallest delay step.

![Figure 3a. Schematic of the Phase Detector.](image-url)
Figure 3b. Layout of the Phase Detector.
Clock Divider

Divides the clock by 8 times so that the counter doesn’t operate at every clock signal. This is done to ensure that the next time the Phase Detector is questioned about the state of the synchronization, the new delay value will have been calculated, applied and allowed to settle. Running the counter at a lower frequency also saves power.

The sizing for the Clock Divider and the Counter was similar. They had no need to be very fast just functionally correct. Therefore we used the same sizing as we had in the laboration.

Figure 4a. Schematic view of the Clock Divider. Special treatment is done for the static enable and reset signals which are not used.
Figure 4b. Layout of the Clock Divider. Since it is 8-bit, it has 3 cells.
Counter

A 5-bit counter that keeps track of the current delay. Depending on if the Phase Detector reports that the delayed clock signal is before or after the phase in the original clock the Counter increases or decreases the delay to compensate. The enable signal was implemented in case we had time to create a lock detector making it possible to put the counter in standby mode. It’s not used and the counter is enabled all the time.

The Counter is built with 5 cells, where each cell has the design as shown in figure 5a below.

![Figure 5a. Block view of a Counter cell.](image-url)
Figure 5b. Schematic view of the DFF used in the Counter’s and the Clock Divider’s TFF.
Figure 5c. Layout of the counter.
Delay Line

Delays the input clock in 16 steps according to the output from the Counter. This results in 16 clock signals, all delayed the same amount. The Delay Line is implemented with a series of inverters coupled with five controllable polycaps. The polycaps have been weighted to make the increase as linear as possible but because we didn’t want to use too big polycaps there is unfortunately a drop in the delay when the control signals are increased from 15 to 16. The delay in the layouted version is considerably larger than that of the schematic, see figure 6.

Figure 6. The delay in a delay line cell for each counter value on schematic cell and full layout level. Simulated in typical mode and 70 degrees. There is a decrease in delay from 15 to 16 and smaller dips from 7 to 8 and 23 to 24.
The first four inverters are sized to give a good inertial delay and the last one is big enough to drive all the transmission gates when they are turned on. The first transmission gate has no polycap because it gives a good delay by itself, just by turning it on and off. The other transmission gates are sized to be as fast as possible as to not be the major part of the delay, but rather that the polycaps stands for the most delay. This was done because polycaps have less variance in fabrication than transistors.
The polycaps have been placed so that a pair of inverters fit in between to act as a driver for the outputs on the long MET4 wires connected to the Transition Detector. Great care was taken to make the output wires of equal length (they differ with max 1 µm and the wire length is 225 µm) to avoid that any of the delayed signals get skewed compared to the others. The outputs are also shielded.
The counter control signals need inverted versions as well. 2 sets of 5 pairs of quite big inverters are used to create the inverted signals as well as act as drivers. 2 sets are used because it was much easier routing and it didn’t occupy much more space.
Transition Detector

Outputs short negative pulses when the inputs from the Delay Line have a rising edge. The pulses are non-overlapping. The Transition Detector is made up by 16 identical cells. Each cell creates pulses by connecting the output from the Delay Line to an AND gate together with the same signal inverted and delayed by sending it through three inverters. Each signal is also buffered before the AND gate.

The transistor sizes have been very carefully tweaked to produce a pulse length of exactly the length required by the Edge Combiner (in TM, WS and WP process corners).

Figure 9a. Schematic view of a Transition Detector cell.

Figure 9b. Layout view of a Transition Detector cell.
Edge Combiner

Generates the output clock for the entire DLL by toggling the output signal every time a pulse arrives from the Transition Detector. This is achieved with a tree of AND gates merging the 16 signals two by two and feeding the merged signal through a T flip-flop that turns it into the final output signal. In order to not lose any pulses in the AND tree due to pulse overlap, the pulses are paired in such a way that they are as far apart as possible (so that each combined signal contains equally distributed pulses).

The Transition Detector combined with the Edge Combiner make sure the output has a duty cycle close to 50% regardless of the duty cycle of the input signal, since only rising edges are considered.

Figure 10a. Schematic view of the Edge Combiner. The last (rightmost) block is the TPL.
Figure 10b. Layout view of the Edge Combiner. Notice how the last NAND-gate is rotated and positioned between the two AND-gates of the previous stage.

**Symmetric AND**

The AND gates are symmetric to make the propagation delay from both inputs equal. The NAND gates and inverters are sized a bit larger to make them as fast enough maintain a clean pulse through the entire tree.
TPL

The TPL is a toggle-pulse latch. It’s basically a 3-inverter ring oscillator with a transmission gate in the loop. When getting the correct pulse lengths as input it acts as a normal T-flip-flop. If given a pulse that is too long the TPL will toggle between 0 and 1 at full speed. This makes it fast enough to output a 1 GHz signal, but it also means that the input pulse length needs to be carefully tuned.

Figure 11a. Schematic view of the TPL
Figure 11b. Layout view of the TPL.
Simulation Results

We have done seven final simulations for our DLL with buffers.

<table>
<thead>
<tr>
<th>Simulation nr</th>
<th>Corner</th>
<th>Input frequency</th>
<th>Temperature</th>
<th>Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cmostm</td>
<td>125 MHz</td>
<td>70</td>
<td>3-5</td>
</tr>
<tr>
<td>2</td>
<td>cmostm</td>
<td>125 MHz</td>
<td>25</td>
<td>13-14</td>
</tr>
<tr>
<td>3</td>
<td>cmosws</td>
<td>83 MHz</td>
<td>70</td>
<td>5-7</td>
</tr>
<tr>
<td>4</td>
<td>cmoswp</td>
<td>142 MHz</td>
<td>70</td>
<td>no lock under 16</td>
</tr>
<tr>
<td>5</td>
<td>cmoswp</td>
<td>150 MHz</td>
<td>25</td>
<td>no lock under 16</td>
</tr>
<tr>
<td>6</td>
<td>cmoswp</td>
<td>140 MHz</td>
<td>70</td>
<td>no lock</td>
</tr>
<tr>
<td>7</td>
<td>cmoswz</td>
<td>125 MHz</td>
<td>70</td>
<td>9-11</td>
</tr>
</tbody>
</table>

According to simulations 1 and 2, the temperature has a big effect on the delay of the delay stages.

Some locks (those of simulations 1, 3 and 7) do not achieve a perfect lock and counts two times up and then two times down, instead of just once as is the case in simulation 2. This is because 1 delay value aligns the delayed clock with the clock signal so good that the PD can’t detect this small difference.
Figure 12. Simulation run 1. The orange graph shows the up/down signal toggling between up and down.
Figure 13. Simulation run 6. The DLL fails to lock at 31. Although it should count down it fails because it isn’t able to see the small difference of 15 ps. Hence the counter overflows to 0.
PAD Assignments / Early Test Plan

Pins

We have 12 pins available, and our pin configuration is as follows:

- 6 input pins
  - Vdd
  - Vdd for the Delay Line
  - Vdd for the Transition Detector
  - Vss
  - Reset (for the counter)
  - Input clock

- 4 output pins
  - Delayed clock, the other input to the Phase Detector
  - Resulting down/up signal from the Phase Detector
  - One of the pulses from the Transition Detector, p15
  - Output clock

This configuration uses 10 pins. The output clock is our main output. The other three outputs are test pins. With the down/up signal we can see if the DLL has achieved a lock (signal switching) or not (signal constantly 0 or 1).

The delayed clock signal shows how the clock has propagated through the Delay Line and the pulse signal will show if the Transition Detector delivers a pulse or not. Because of the short width of the pulse we will probably not be able to measure its width.
Risks and Delays

There is a risk that the delay produced by the delay line is not long enough. Another risk is that the pulses from the transition detector doesn’t have the correct width. We plan to cope with both these risks by introducing separate Vdd:s for each of them.

The wires in the layout introduced capacitance and resistance that weren’t in the schematics. We mitigated this problem by adding several interconnection drivers.

We initially planned for a configurable multiplier to address potential duty cycle problems relating to the long Delay Line. But when simulations suggested that this wasn’t an issue (possibly because the Delay Line itself was designed with this problem in mind) this configurability was dropped in favor of more time dedicated to tuning the 8x mode.
Project Evaluation

Very good supervisor!

Lagom amount of work.

Good laborations suited to make the project easier. Some additional virtuoso commands could be introduced, like reshape (R) and edit in place (x). A nice schematic command is M. Show the alternative result browser (graph viewer).

References

1. Jin-Han Kim, Young-Ho Kwak, Seok-Ryung Yoon, Moo-Young Kim, Soo-Won Kim, Chulwoo Kim, A CMOS DLL-Based 120MHz to 1.8GHz Clock Generator for Dynamic Frequency Scaling, ISSCC 2005, Korea University, Seoul, Korea