CMOS Project Report
Delay Measurement Device

Nathan Dwek – Ilias Fassi Fihri

May 31, 2016

Contents

1 Introduction

1.1 Specifications

The goal of this project is to design and simulate a delay measurement device using components available through the AMS C35 0.35 µm CMOS process. In order to measure the length of a long cable, a voltage pulse is applied to one end of the cable and the length is deduced from the time it takes to reach the other end.

The output of the device must scale with the delay between two voltage pulses, which can range between 10 ns and 500 ns. The pulses themselves are chosen to be 10 ns long, and the device must be able to drive a 10 pF capacitive load.

1.2 Basic Block Diagram

To achieve this, the device is organized as shown in figure 1.

Figure 1: Block diagram overview.

At the core is the block $\boxed{\texttt{rampgen}}$ which generates a rising ramp on its analog output `measure` when the digital input `charge` is *`LO`*, maintains its output when `charge` is *`HI`*, and resets it to ground when the digital input `start` is *`HI`*.

It is the controlled by the first block, $\boxed{\texttt{logic}}$, which controls `charge` based on the succession of pulses on its digital inputs `start` and `stop` and some internal logic.

The output of $\boxed{\texttt{rampgen}}$ is buffered by the output stage $\boxed{\texttt{buffer}}$ which is able to drive the specified load of 10 pF.

In the next three sections, the design of each of these blocks is reviewed. Then, in the last section, the operation curves of the whole are presented.

2 Internal logic

2.1 Derivation of a Truth Table

First, we notice that the system needs to have a memory. Indeed, after a pulse has occured, and when both inputs are *`LO`*, the output `charge` can still be *`HI`* or *`LO`* depending on the last pulse. Given the logic that needs to be implemented and the fact that there are two separate pulse inputs we choose to implement the memory using an RS-latch. This choice will prove to be adequate at the end of this section. Intuitively, we choose to put `start` on the `set` of the latch and `stop` on the `reset`.

Furthermore, we cannot immediately reset `measure` to ground upon receiving a pulse on `stop`, because this wouldn't give enough time for the output stage to properly latch the final value[1], or more generally, wouldn't leave time for any reading device to use the measurement. For this reason, we choose to reset the whole device at the rising edge of a pulse on `start`, and to start the measurement at the falling edge of the pulse. To correctly measure the delay, the measurement must then be stopped at the falling edge of the pulse on `stop`. The output voltage is thus stabilized at its final value and available for reading from the end of the `stop` pulse until the beginning of the next `start` pulse.

With that it mind, it is possible to derive a first logical function, where `Q` is the output of the RS-latch:

$$\mathtt{charge}(\mathtt{start}, \mathtt{stop}, \mathtt{Q}) = \mathtt{Q} \cdot \overline{\mathtt{start}} + \mathtt{stop} \tag{1}$$

2.2 NAND/NOR **Implementation**

This function must be adapted based on the following implementation details:

- As will be shown in next section, `charge` should be active-low because it drives a PMOS switch.
- NAND and NOR gates are the most directly available and require less transistors than AND and OR gates.
- $\overline{\mathtt{Q}}$ is directly available from the RS-latch.

Using de Morgan's law on the first term of the NOR which appears when taking account that `charge` should be active low, we find:

$$\begin{aligned} \overline{\mathtt{charge}} &= \mathrm{NOR}(\mathtt{Q} \cdot \overline{\mathtt{start}}, \mathtt{stop}) \\ &= \mathrm{NOR}(\mathrm{NOR}(\overline{\mathtt{Q}}, \mathtt{start}), \mathtt{stop}) \end{aligned}$$

This final expression allows to implement the required logic using only one RS-latch and two NOR gates (rather than for example four NOT, one NAND and one NOR gate if we simply inverted function 1), as shown on figure 2.

[1] In our case the output stage doesn't even have a memory so this doesn't really apply.

Figure 2: Logic circuit of `logic` .

2.3 Sizing

The logical gates are built using minimal length transistors. The width should theoretically progressively increase from input gates to output gates so that the output gate is able to drive the parasitic input capacitance of the next block and so that every gate in the circuit is able to drive the gate at its output. However, it turns out that minimally-sized transistors are able to correctly drive `rampgen` , so the whole logic circuit is made of minimally-size transistors.