50. UKW-Tagung Weinheim

Increasing the resolution of reciprocal frequency counters

Paul Boven, PE1NUT, p.boven@xs4all.nl

1 Introduction

The frequency counter is one of the simplest electronic measurement devices. It counts the number of cycles of the input signal during a set interval, and can directly display this result as the frequency of the input signal. The accuracy of this measurement is first of all determined by the quality of the counter's reference oscillator. From simple crystals to atomic frequency standards, there is a wide range of solutions to choose from to achieve any desired precision. But how to build or obtain a suitable reference has been the subject of several publications already, and will not be covered here.

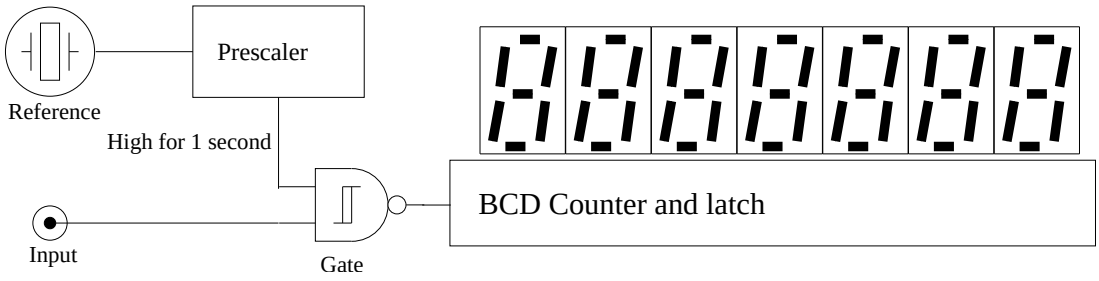

Figure 1: A very simple frequency counter

Another key parameter of a frequency counter is the resolution it achieves. A simple counter that counts cycles for one second would have a resolution of 1 Hz, measuring for 10 seconds would increase that to 0.1 Hz and so on. So the resolution, too, can be chosen almost arbitrarily given enough time. Resolution in itself is therefore not a good indication of the performance of a counter, it should always be considered in relation to the measurement duration.

This paper describes some designs that can achieve much higher resolution in less time than the basic frequency counter. Also shown is a design that can be easily implemented into a single, cheap FPGA chip.

2 Normal and reciprocal counters

The resolution of the basic frequency counter is limited by the fact that it can only count complete cycles of the input signal. The measurement will start at some random phase of the input signal and will stop randomly somewhere during a cycle of the input, too. There is no way for the counter to determine how much of a cycle it missed at the beginning and end of the measurement. It could have missed almost a complete cycle at both the beginning and end, or only a very small bit. Hence the resolution for a 1 second measurement interval will be ± 1 Hz.

2.1 The reciprocal counter

By swapping the roles of input signal and reference, it becomes possible to create a counter that starts and stops the measurement precisely in step with the cycles of the unknown input signal. During the measurement it now counts the number of cycles of the reference. In a regular frequency counter, the reference will be divided down to a low frequency of e.g. 1 Hz. In this reversed counter the reference should be as high as possible because this determines the resolution of the measurement. With a reference of e.g. 1 GHz, the measurement corresponds to the nummer of 1 ns cycles of the reference that occur during one period of the input signal. This is in fact a measurement of the duration of one cycle of the input signal, its period t_x. The frequency f_x can then be computed by calculating the reciprocal $f_x = 1/t_x$. This class of counters is called 'reciprocal counter' because of this calculation. A reciprocal counter needs an arithmatic unit (e.g. a microprocessor) to convert from period to frequency.

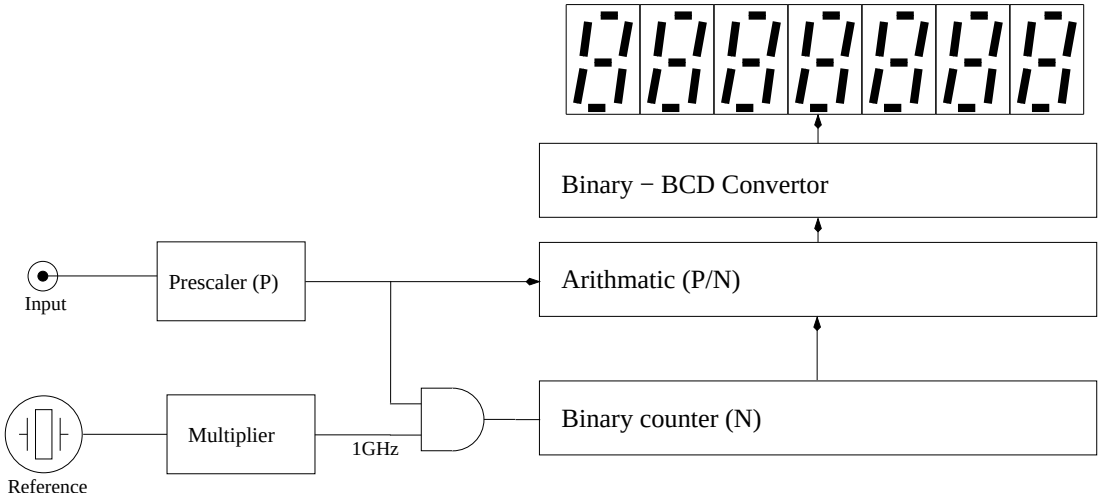

Figure 2: A simplified reciprocal counter

In the reciprocal counter, the duration of the measurement is determined by the input signal. If the measurement only were to last a single cycle of the input, it would get

rather short for higher frequencies. This is why a prescaler is inserted before the counting section. Modern reciprocal counters set the prescaler automatically in accordance with the resolution or measurement interval specified by the user.

2.2 Comparing resolutions

Because the reciprocal counter can only count complete cycles of the reference, for a 1 GHz reference the resulting error in the measurement would be ± 1 ns. Under the assumption that the prescaler is set such that the measurement duration is about 1 second (the output of the prescaler $\approx$ 1 Hz), the relative resolution would be 1 ns / 1 s = 10^{-9}. This is independent of the input signal. The resolution of a frequency counter is simply the reference frequency divided by the measurement interval. And the higher the reference, the better the performance of the counter.

With a regular counter, the same type of error amounts to plus or minus one cycle of the input signal. In the frequency domain that equals ± 1 Hz (or whatever the measurement interval is), in the time domain the relation would be $\Delta t_x = \pm 1/f_x$. The relative resolution for this type of counter becomes rather poor at low frequencies.

Usually the limiting factor in a frequency counter is how high a frequency the first flipflop in the counter can handle. Both regular and reciprocal counters are equally affected by this limitation. When building a regular and a reciprocal frequency counter using the same kind of technology, the reciprocal counter would always hold the advantage when its reference is made equal to the highest frequency that can be handled. This can easily be understood by realising that one cycle of the input signal in a regular counter will be longer than one cycle of the high frequency reference in a reciprocal counter.

3 Interpolation

Increasing the reference frequency in a reciprocal counter would improve its resolution, but it would also make the design of the counter much more complicated and expensive. Because the reference is a known and fixed frequency, it turns out to be possible to gain resolution by interpolation of the reference phase.

3.1 Analog interpolation

Counters using interpolation became widely used in the 1960's and 1970's. These would charge a small capacitor in the very small interval between an edge of the input signal and the start of the next reference cycle. The charge stored is directly proportional to the length of the remainder of the reference cycle. This charge can then be measured by counting how long it takes to discharge the capacitor with a much smaller current. The gain in resolution is theoretically equal to the ratio of the charging and discharging current.

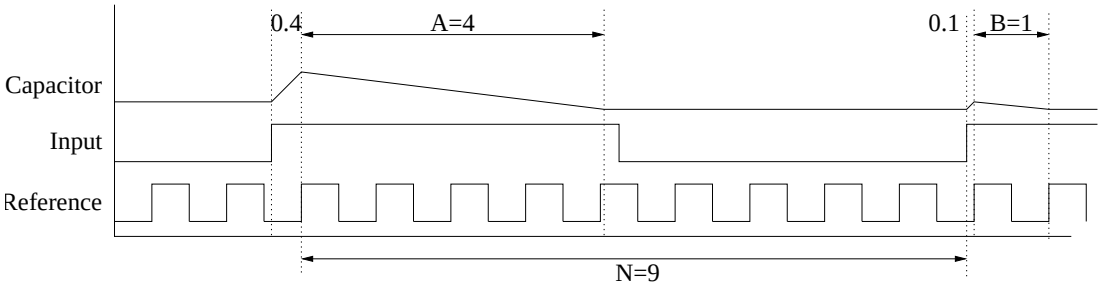

Figure 3: Analog interpolation

In the example above, the main counter N sees 9 cycles of the reference between the two rising edges of the input signal. The start of the measurement is 0.4 cycle early compared to the reference. In this example the capacitor discharges 10 times as slow as it charges, so the gain of the circuit is $Q = 10$. The discharge-counter reads 4 at the start of the measurement, and 1 at the end. The period of the input signal is then given by $t_x = t_{ref} * (N + (A - B)/Q)$. In this example the period of the input signal is 9.3 times that of the reference.

3.2 Digital interpolation

In the early 1980's, Hewlett-Packard introduced the HP5370 time interval counter. It uses an internal reference of 200 MHz, but the resoltion is improved 256 times by digital interpolation. This would be equivalent to a 50 GHz reference, and a resolution of only 20ps. Going beyond that is hardly usefull because the jitter in the input circuits is also of that magnitude.

The HP5370 achieves its interpolation by using an oscillator that runs at 255/256th of the reference. It's kept at this precise frequency by a PLL circuit. But at the start of a measurement, it gets restarted as soon as the rising edge of the input signal comes along. The PSPLO (Phase Startable Phase Lockable Oscillator) then keeps running with the old frequency, but a different phase. Because the frequency is $(1-1/256)f_{ref}$, it will coincide with the reference oscillator once every 255 reference cycles. The number of cycles between a restart and the next coincidence is in fact a direct measurement of the fraction of the reference cycle at the restart of the oscillator. This can be compared to the way the vernier on a caliper increases the resolution of the readout.

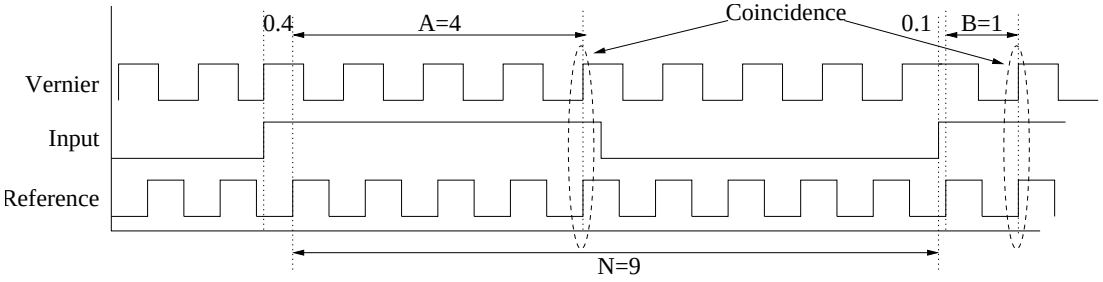

Figure 4: Digital interpolation

In the example above, the period of the Vernier signal is 1.1 times the Reference, giving once again rise to a 10-fold increase in resolution. The other signals and timings are the same as in the previous example. Note where the Vernier oscillator gets restarted, and when the coincidences with the Reference occur. The manual and schematics of the HP5370 are freely available on the Agilent website and well worth studying.

4 FPGAs

Building any of the designs discussed so far requires quite a lot of digital logic, running at high speed. The design of the circuit, creating a suitable PCB layout and even getting hold of the necessary parts can be quite a challenge for HAMs and other hobbyists. This is where FPGAs come to the rescue. These are integrated circuits that contain a large number of logic elements and flipflops, that can all be freely configured by the end-user. This is almost like having your very own chip factory at home. And FPGAs can be reconfigured at will without having to remove the IC from the circuit. The number of logic gates in an FPGA can run from several thousand to several million. This is an enormous capacity: the bigger FPGAs can hold several PowerPC cpu's, or a complete home-computer such as the Comodore-64, on a single chip. Modern FPGAs can handle signals of up to several hundred megahertz and because they are completely parallel in nature, they can process very large amounts of data. The number of pins on a single FPGA can be well over a thousand.

4.1 Working with FPGAs

The larger FPGAs, with over 1000 pins, are a bit beyond what an amateur could hope to succesfully solder onto a PCB. But the smaller ones are available in TQFP packages, with e.g. 25 leads on every side of a square for a pincount of 100. The distance between leads is then 0.5 mm which requires a very small soldering iron and a steady hand, but is just doable. And the resources in these small devices are still considerable: the Xilinx Spartan 3 that was used for this project contains 200k user configurable system gates. It currently costs US$ 13.45 in single item quantities, directly from the manufacturer's website.

The software needed to design with FPGAs is made available for free by the manufacturer, and runs both under Windows or Linux. The free software from Xilinx only supports the smaller devices, but those are the devices most usefull to HAMs anyway. It supports schematic entry (very convenient for beginners) and the special hardware description languages such as VHDL or Verilog. These languages make it possible to efficiently describe a circuit, and the software then takes care of translating the design into a bit-stream that can be loaded into the device.

A good way to get started in FPGAs is by buying a starter kit. These come with an FPGA already mounted on a PCB, with all kinds of switches, leds and connections to

play around with. The Xilinx Spartan-3 starter kit that is used for this project sells for US$ 99 and comes with an XC3S200 FPGA (200.000 logic gates), 1 MB of RAM, a 50 MHz oscillator, connectors for RS232, VGA and PS/2, 8 leds, 12 switches and a 4 digit 7 segment LED display. Software and a cable for connecting the starter-kit to a parallel port are also included.

4.2 Spartan-3 internals

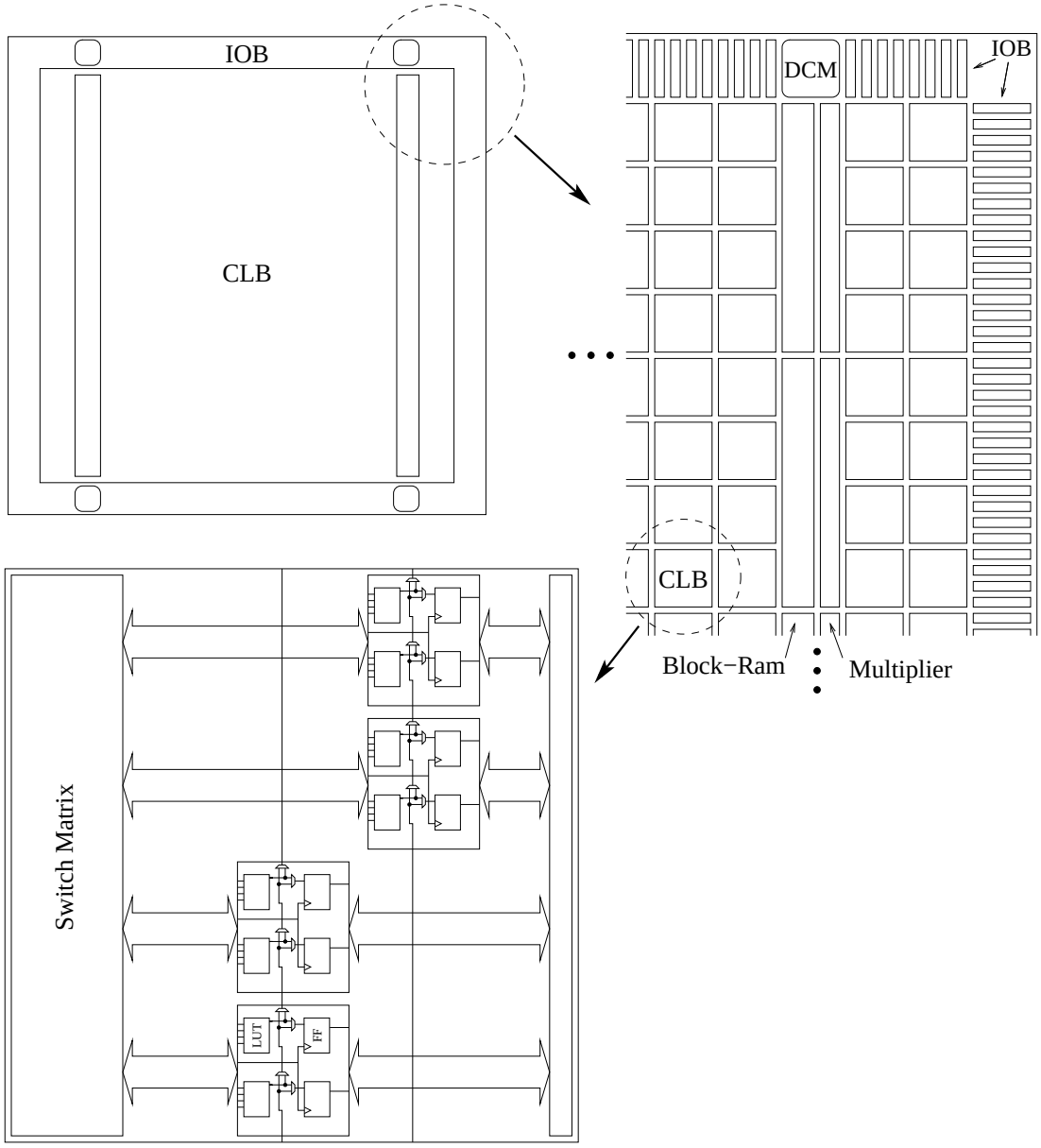

Figure 5: A Spartan-3 FPGA

The Spartan-3 consists of the following parts:

IOB Input Output Blocks, these connect to the pins of the FPGA package.

CLB Configurable Logic Blocks, which contain all the configurable logic of the FPGA

LUT Look-Up Table, this is a sub-part of the CLB and is a small table that can act as any logic function with 4 inputs and 1 output.

FF Flip-Flop, like LUTs each CLB has 8 of these.

DCM Digital Clock Manager, for controlling clock-signals

The CLBs are aranged in a rectangular pattern that spans most of the die of the chip, with IOBs running along the perimeter. In between all the CLBs and IOBs are the pathways that allow these sections to connect to all other locations on the chip. Two columns hold special purpose hardware: block-ram for storing data, and multipliers that can quickly perform the arithmetic of multiplication, which can be especially usefull when using the FPGA for Digital Signal Processing.

4.3 DCM

There are four Digital Clock Manager circuits on most of the Xilinx Spartan-3 FPGAs, situated along the top and bottom edge of the chip. These can be used for multiplying, dividing or phase-shifting clock signals before they are distributed over the chip. A DCM consists of a large number of delay elements in a row. The clock signal is fed into the first delay, and the DCM determines how much delay constitutes a complete cycle of the input clock. It then creates a feedback from the output of the delay element in question back to the input, creating a Delay Locked Loop (DLL) to copy and condition the input clock signal. The DCM also determines which delay outputs correspond to a phase shift of 90, 180 and 270 degrees, and these are made available as outputs as well. The DCM is constantly adjusting which delay tap to use to keep the generated signal in phase with the input signal. Switching between the delay outputs causes a jitter of aproximately 100ps.

5 A single chip reciprocal counter

When driving a DCM with a 100 MHz clock signal, it will output four copies of the 100 MHz clock, each 90 degrees (2.5 ns) apart. These four clock signals make it possible to determine the relationship between input signal and reference with 2.5 ns resolution, a fourfold gain. By using all four DCMs, each with a 22.5 degree phase shift in relation to its neighbour, it is possible to cover a reference cycle of 100MHz in increments of 625ps for a virtual reference of 1.6GHz.

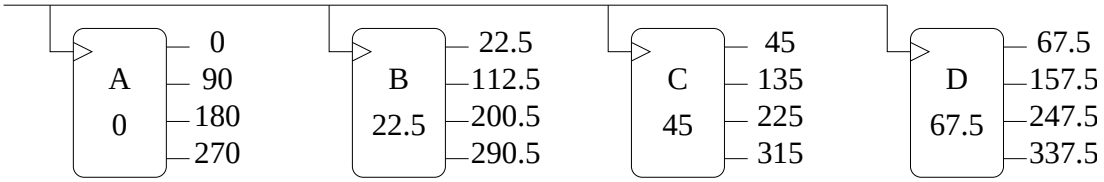

Figure 6: Using 4 DCMs, each offset $22.5°$

Using all 4 DCMs makes it possible to increase the resolution by a factor of 16, even though the highest frequency used in the circuit remains 100 MHz. It is important to realise that this extra resolution does not neccesarily correspond to an equally high maximum input frequency. Using a small external prescaler can improve this without adversely impacting the performance of the counter, which will need to further scale down the input signal anyway.

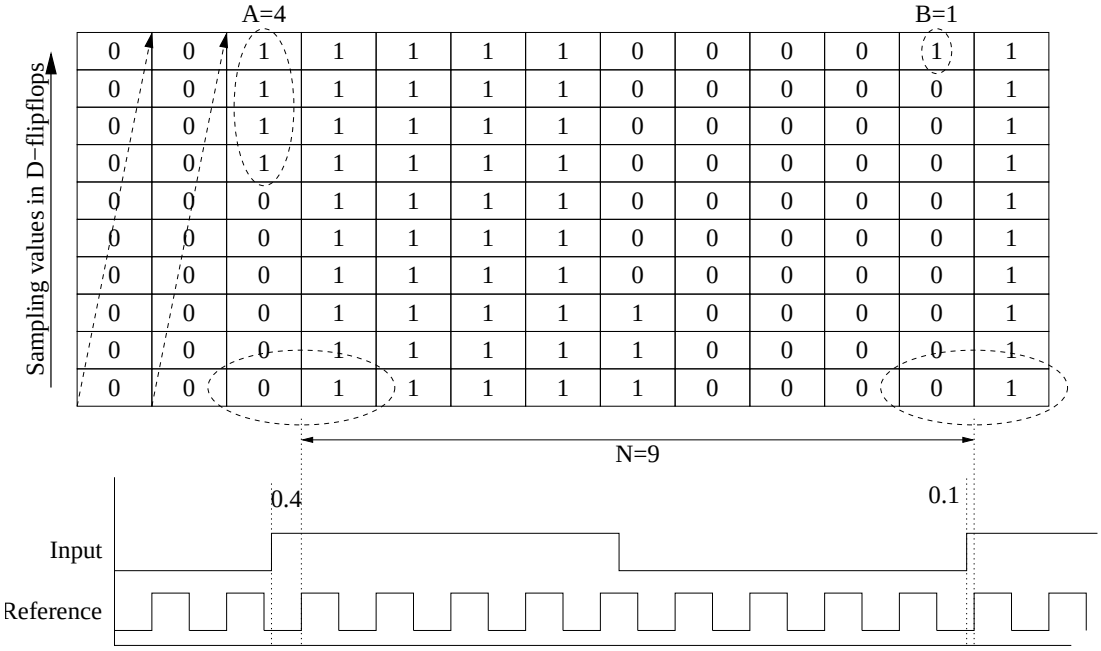

Figure 7: Interpolation with DCMs

The (prescaled) input signal can be sampled by using all the 16 outputs of the 4 DCMs to each trigger a D-flipflop. (The example above depicts this situation but with a gain of 10, to be comparable to the earlier examples). The flipflops are triggered 625ps after one another. A rising edge of the input signal will cause some of the flipflops to still register a low, while all flipflops that get triggered after the rising edge register high. The 16 bit output pattern can be easily converted to a 4 bit binary representation, these 4 bits also correspond to the extra resolution the counter has gained.

The input signal is also used to sample the value of the continuously running counter N. The value of that counter, together with the 4 bits of extra resolution as least significant bits, provide a binary timestamp. The difference with the previous (stored) timestamp is the actual period of the signal being measured. Finally, after

scaling this value and performing the reciprocal operation, the measurement can be converted to BCD and displayed.

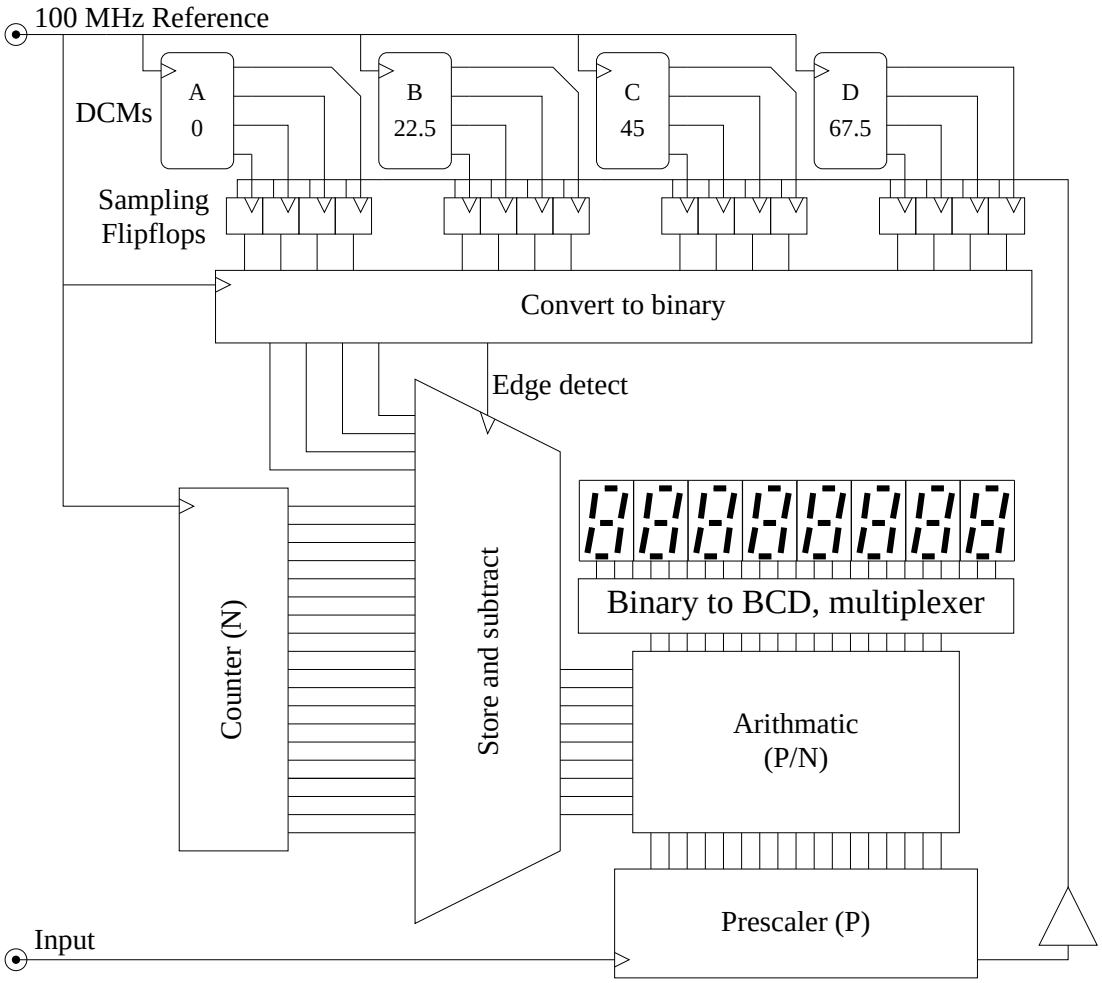

Figure 8: An interpolating reciprocal counter in FPGA

The pathways within the FPGA can cause delays of several ns. To make the delays from the DCMs to the Sampling flipflops as low as possible, the flipflops must be placed close to their respective DCMs. The positioning of the time-critical elements within the PCB can be quite a puzzle, but the majority of the circuit is not very time-critical and placement can be left to the software.

This design is still a work-in-progress. It only takes up a small part of the FPGA, leaving room for further extensions. Improvements might be achieved by changing the reference frequency to 150MHz (the phase-shifting of the DCMs works up to 165MHz) and by using the propagation delays within the FPGA to further increase the timing resolution.

All the design and testing so far has been done on the Spartan-3 development kit. The final stage of this project will be designing a PCB to hold an FPGA, a frequency reference and input stages, to turn this into a stand-alone counter.