Ratio Detector

The ratio detector is a fairly simple circuit. It is displayed in the figure below, and is nothing new. It's simply an event counter – it counts how many time event A occurs over a period of N event Bs. When you think about it, it's exactly what we want. We have input frames (left/right samples) occurring at the sample rate, and 64 clocks per output sample frame, so we have complete information regarding the output to input sample rate ratio.

The lower register counts up every input frame sync pulse to a terminal value, which triggers another register to latch the value of a counter that increments every time an output clock event occurs. The following register is only enabled upon terminal count, and if its output is greater or less than the value of the Fso count register. In this way we implement what is known as hysteresis – which is a fancy term for the fact that there is some slack in the value, so that only significant changes in the input give rise to a change in the output. At the output of this register we have the input to output sample rate ratio with an integer range of 0 to 15.

Following on from this, we apply the ratio to a very simple low pass filter. Actually, this filter is what is known as a leaky integrator (I've never questioned why it is called leaky, Google probably knows why). The integrator output slowly changes with response to an input stimulus until the input and output are equal in value. How does it work I hear you ask? Conceptually it is very simple! Firstly, to understand what the circuit does, we need to understand what integration is. Integration is a mathematical concept where, given a continuous line with X and Y coordinates, we calculate the area under the line. In discrete time systems, this amounts to continually adding values of a signal to itself. This is achieved by adding a register output to its input – producing an error term. Sounds simple really, and it is.

Now to explain the circuit. The output of the integrator is subtracted from the input signal – producing an error term. This error term is added to the output of the integrator (this is how it performs integration) which is registered to be output on the next clock cycle. What does this achieve? Well, in terms of the ratio detector, we are producing a signal that is constant, and so we expect the integrator to slowly rise or fall from its current value to the value presented to its input. How fast it does this is dependent on its programmed gain, and the result is very much like a simple RC low pass filter – which is because just like an RC low pass filter, it is a first order filter. So, by utilising this integrator, we only allow the output to change slowly with any given input.

Finally, the output of the integrator is presented to a limiter. Earlier I mentioned that the calculated ratio could have an integer value in the range of 0 to 15, well the limiter restricts the total range of the ratio output to values between zero and one. The fractional portion is critically important, and I'll explain that later when I discuss the SRC algorithm, but form now, it's only important to know that the output of the ratio detector cannot be greater than one.

Now, I should mention that whilst none of these components are novel in and of themselves, they were first presented in this format in US patent 5471411 back in 1995. At least some of the inventors worked for Analog Devices at the time this patent was granted, and maybe still do. You may have heard of the AD1896? I believe that, at least from what I can tell from the datasheet, this circuit is still in use. Thanks go to Adams, Kwan, and Coln for this work.

Ramp Filter

Whilst the ratio detector was fairly simple and straight forward, the ramp filter is a different beast all together. Firstly, I should acknowledge Adams, Kwan, and Coln again for this – it's their invention, and I've used it because I believe that it is the best solution for the problem at hand.

But what is the problem at hand? Well, the SRC has sample arriving at a fixed rate, and we want to generate output samples based on these input samples, but with totally new timing. So, in order to do that, we need some way of knowing which input samples we need to use in order to calculate an output sample, and how these input samples relate to the output with respect to timing. The way we do this is with a ramp filter.

This component produces an output signal that is, in effect, a ramp. If you were to draw the output of the ramp filter it on a graph, it would start at some value, and would climb in a straight line, continually going positive, to an infinity. In digital terms, it will wrap - meaning roll over to zero and start again at some point. It is a filter because it doesn't change quickly with respect to its input – i.e. it attenuates or rejects sudden variations in its input from making it to the output.

Now, what we ramp is the current number of samples we have received at the time an output sample is requested. Remembering that the input and output sample rates don't have to have any relationship to each other (asynchronous), you can see that we could have a fractional number of input samples in our buffer when an output sample is requested – of course the fractional samples don't exist, but this information is critical for producing a good quality output.

Lock Detection

Ratio Detector Problem

So, the problem here is, how do we know when the SRC has obtained lock? What does it mean to have lock? What are we locking to?

Well the latter two parts of the problem are fairly easy to explain. Simply put, we have lock when the output sample rate to input sampling rate detector becomes stable. A fairly useless explanation in the sense that we haven't defined stability, so we'll describe what that means first. Here's out stability criteria:

1. Input and output clocks should be present. With respect to the input clock, we actually only need the left/right strobe signal – but the SRC is designed so that you can use either by synthesis configuration;
2. The output to input sample rate ratio should be greater than 0.125. Why? Because we simply don't have enough RAM available internally to accommodate down-sampling by a factor greater than 7.5:1 – and although 0.125 implies down-sampling by 8, it's close enough (and simple enough to implement);
3. The first derivative of the input to output of the integrator used for ratio tracking should be close to zero. This might require some explaining, and actually, the explanation is probably more difficult that the actual implementation.

To explain point 3 above, let's consider how the integrator behaves. Below is an image of the output of the integrator.

FIGURE

When unchanging input and output clocks are present the integrator is presented with a constant input. When the integrator has reached steady state, its output should reflect the input precisely. In other words, the output should eventually become constant. Now, the derivative of the output simply looks at its rate of change over time – and since it is constant, this should settle to zero (and will therefore be non-zero when I/o clocks change).

In mathematical terms, under steady state conditions, we can see that

f(x) = C

Where x is the ratio function. Therefore, taking the derivative of f(x) we can see that:

dx/dt = 0

Implementing such a thing is really quite simple. We simply subtract the input of the integrator from its output (i.e. the rate of change of its input and output) and look at the absolute value of the result. This gives us positive value to compare against a threshold for ratio lock.

Ramp Filter Problem

This is only part of the picture though. Remember, we have a ramp filter (or what Adams calls an auto-centring circuit) to consider too, and if this filter hasn't reached steady state then it's possible that we'll hear what is known as a doppler of the input signal.

So, we ideally want the ramp filter to have settled to some value too. And this makes sense when we consider that the output of the ramp filter provides critical timing information, including:

1. The initial address for RAM lookup (our ring buffer into which input samples are stored);
2. The initial address for our filter coefficient ROM; and
3. The interpolation factor that we use to interpolate the filter.

So having the ramp filter settle is really critical. It can be seen that the ramp filter has a fast response that we use to coarsely track to our intended steady state, and a slow response so that the filter doesn't respond overly to perturbations in its input (i.e. jitter). The selection of fast versus slow filter response is controlled by the ratio lock detector outlined above, but once in slow mode, how can we tell if the filter has reached steady state?

The answer is similar to the method used in the ratio lock detector – except this time we look at the second derivative of the ramp filter output. Why the second derivative? Well, remember from above that if a signal is unchanging at steady state, then the derivative of that signal will be zero. In this case we have a ramp to consider, and the first derivative of the this signal will be some constant greater than zero – so we take this and call it the first derivative of the ramp output. If we take the second derivative, baring in mind that the first derivative is some non-zero constant, then the result should be zero under steady state conditions.

The maths makes this pretty easy to understand. The formula for a straight line in:

f(x) = mt + C

Where f(x) is the output of the ramp filter, m is some slope of a straight line, and C is some constant. The first derivative, therefore, is:

dx/dt = m

And then, just as describe above, the derivative of this is:

d2x/dt2 = 0

However, we need to consider the fact that if the filter has settled in fast response mode, then the second derivative may have gone to zero at the time of mode switch to slow mode. Switching response modes will upset the ramp filter, so we wait for the second derivative to climb above some threshold, and then approach zero again before we declare that the SRC as a whole has achieved lock. Let's look at how the actual ramp filter behaves in practice.

FIGURE

Here, in zone 1 we can see the ramp filter rise from reset conditions and settle quite quickly. Zone 2 shows the reaction of the filter to the sudden gain change caused by the change from fast response, to slow response. The filter changes to some peak before settling around zero. So, when in slow mode, we detect that the second derivative of the ramp filter has risen above some value and settled again before SRC lock is declared.

When does the SRC leave lock mode?

Simple, if either of the ratio or ramp lock detector derivative values rise above some threshold then we say that we have left lock. Also, should either the input or output sample clocks halt, when we declare the ratio lock detector to have “unlocked” and therefore the SRC to leave locked state.