Power Servoing IP Reference

Use the Power Servoing IP for power amplifier (PA) testing where you send a waveform to a PA device under test (DUT), acquire the output power of the DUT, and adjust the gain of the generated signal to achieve your desired output power.

Table of Contents

Overview

Wireless communication systems increase signal strength with a PA integrated circuit (IC) before sending the signal to the antenna.

PA performance depends on the output power level, so it is important to operate your PA at a specified output power level for testing. However, you typically only know a rough gain range of the PA (for example, ±3 dB). The gain is also non-linear over the operating range of the device—it begins to compress as the output power approaches the maximum. Therefore, you must *level* the output prior to taking any performance measurements. The basic principle of *power leveling* (also known as *power servoing*) is adjusting the power of the signal going into the PA until you reach the desired output power level.

Hardware and Software Requirements

This example uses the PXIe-5840 Vector Signal Transceiver (VST). The example was designed and tested using the Basic Elements instrument design library.

Theory of Operation

Traditional Method

The following figure shows the traditional test setup for measuring a PA. You use a vector signal generator (VSG) to generate a stimulus waveform to the PA DUT, a power meter to ensure the DUT outputs the correct power level, and then a vector signal analyzer (VSA) to measure the performance of the DUT (error vector magnitude (EVM), adjacent channel power (ACP), and so on). Take these measurements for a variety of center frequencies and power levels.

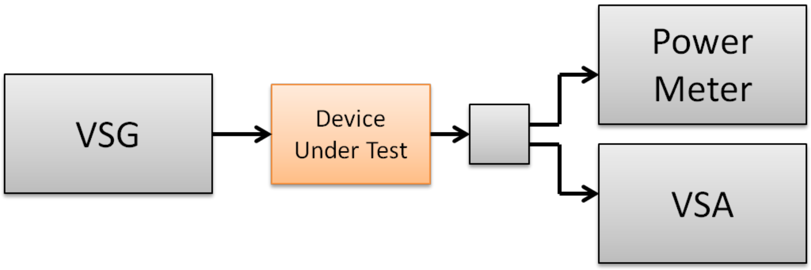


Figure 1. Traditional PA Test Setup

For each desired center frequency and power level, you must level the output power. A typical leveling process executes in the following order:

1. Choose a starting VSG power level based on the estimated gain of the DUT.
2. Set the VSG power level.
3. Wait for the VSG to settle.
4. Wait for the DUT to settle.
5. Take a measurement with the power meter.
6. If power is in range, exit; otherwise, compute a new VSG power level and return to step 2.

This process takes a few hundred milliseconds to several seconds, depending on the type of DUT, the accuracy required, and your instrumentation. After leveling, you can take performance measurements using the VSA.

The following figure shows example output power while leveling a DUT using the traditional method. When applying a stimulus waveform at 0 dBm to a DUT with an expected gain of 28 dB, if the PA gain is linear and matches the typical gain specified in the data sheet, then the very first point generated by the VSG would produce an output power of 28 dBm. Instead, the PA output power is only 26.5 dBm, which demonstrates the inaccuracy of the typical gain specification. Next, the VSG output power is adjusted, and the VSA captures another point. This time the average power is 27.6 dBm, which demonstrates the compression of the PA gain response. In this example, seven steps are required, over about 150 ms, for the PA output to reach the desired power level. In this example, the DUT takes 10 ms to settle in each step. However, settling time can vary from one DUT to the next, which greatly affects the overall leveling time.

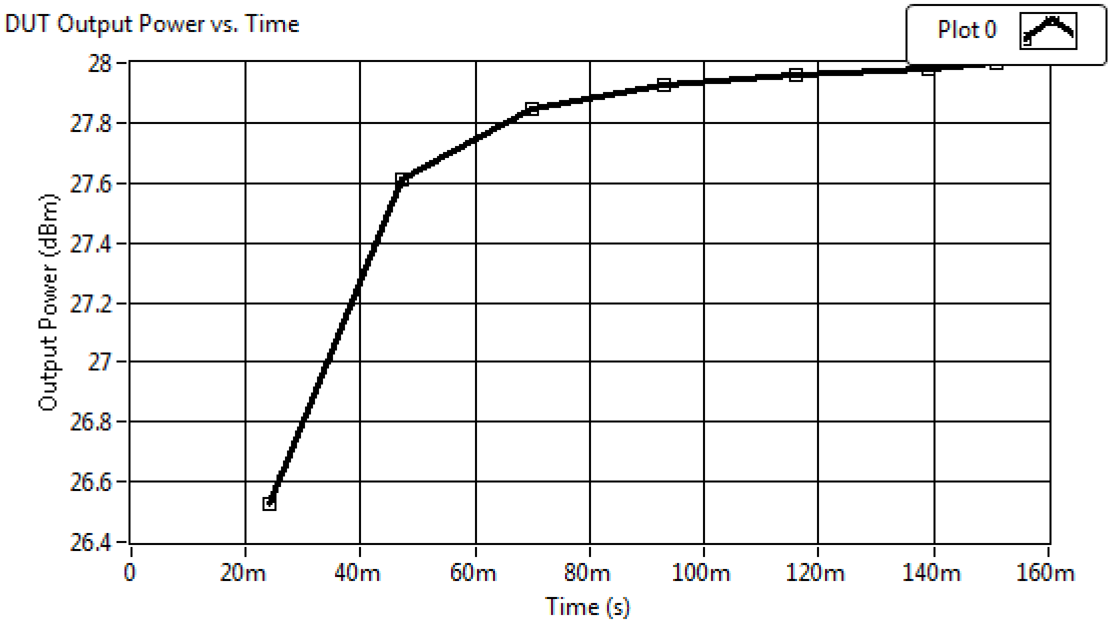


Figure 2. Graph of Traditional Leveling

Hardware-Based Method

The PXIe-5840 combines the features of a VSA, VSG, and user-programmable FPGA. The combination of these features allows you to move the power leveling algorithm into hardware.

Note: Even though there is no power meter shown in the following figure, you typically run a system calibration step to transfer the accuracy of a power meter to the VSA.

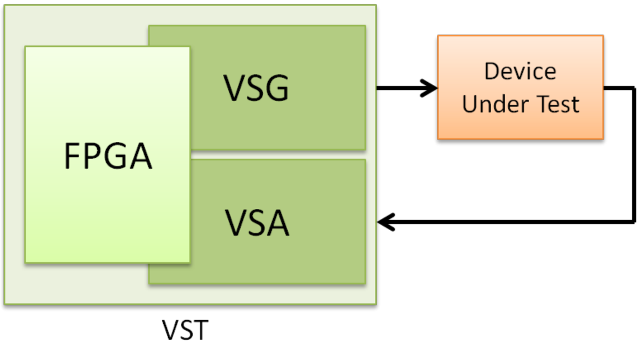


Figure 3. VST PA Test Setup

You can use the hardware-based process to level the PA output power as with the traditional method, but now the leveling loop runs inside the FPGA on the VST. Moving the loop to your hardware greatly reduces the time required to make each adjustment. The following figure shows the FPGA with the Auto Level IP. The Auto Level IP operates in parallel with the rest of the VSA and VSG code (that is, it does not interfere with the VSA and VSG functionality of the instrument). The IP measures the average power the VSA receives and adjusts the power the VSG sends accordingly.

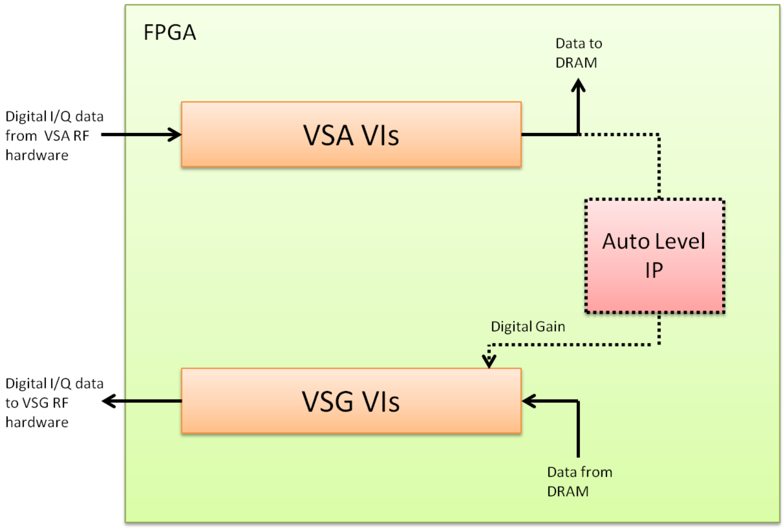

FPGA
Data to DRAM
Digital I/Q data from VSA RF hardware
VSA VIs
Auto Level IP
Digital Gain
Digital I/Q data to VSG RF hardware
VSG VIs
Data from DRAM

Figure 4. VST with Auto Level IP

If you use hardware-based leveling, data is no longer transferred to and from the host computer. Your calculation of VSG output power occurs in real time inside the FPGA, which improves speed. With control theory, you can further accelerate the process by leveling the PA output power while the DUT settles.

The following figure shows the control loop for the traditional leveling process. Each step executes serially, waiting for the previous step to fully settle before continuing. This approach ensures that the loop completes in the fewest number of steps possible, but each step takes time.

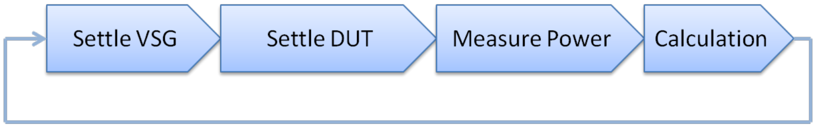


Figure 5. Traditional Leveling Loop

An alternative approach is to repeatedly update the loop while the DUT settles. Each step executes in parallel. This approach may take more steps to complete leveling, but it ensures that the loop completes in the smallest amount of time possible. Because the DUT is still settling while each power measurement is made, the power measurement has some error associated with it. The process still works correctly because the error decreases to zero as the DUT output power converges on a leveled state.

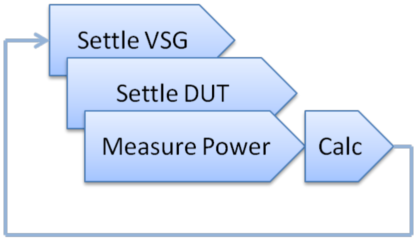


Figure 6. Leveling Loop with Parallel Settling

You can further improve the processing model by changing how you set the VSG output power. Traditionally the front end of a VSG contains amplifier stages, attenuator stages, and switches to select the output power. These take time to set and settle. Avoid this delay by inserting a digital gain control into the digital in-phase and quadrature (I/Q) data path of the VSG and controlling the output power linearly using the digital gain. The actual output power of the VSG is only set once, and the rest of the loop iterations are controlled with the digital gain. This approach provides immediate settling of the VSG output power and a linear response.

You can also reduce the time it takes to measure the output power by employing an averaging schedule. For example, you may know that you need to average about 1 ms of acquired data for a particular Long-Term Evolution (LTE) waveform to get an average power measurement that you can repeat within 0.01 dB accuracy with the VSA. You could choose to average that duration for every iteration of the loop. However, for the first several iterations of the loop, the gain estimate is considerably off from the actual gain of the DUT. An averaging schedule uses shorter averages for the first few iterations of the loop, which executes faster but gets a less accurate reading of the average power, then uses longer averages to get a very accurate reading of the average power as the loop starts to converge. This schedule gets the loop close to leveling quickly and speeds up the overall leveling process. As shown in the following figure, a simple digital gain (DG) adjustment replaces the Settle VSG step, the Measure Power step increases in length as the loop progresses, and the DUT settles in parallel to the leveling loop.

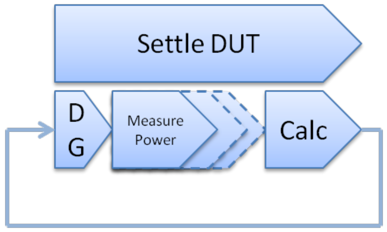


Figure 7. Leveling Loop with Averaging Schedule

With this technique, leveling the DUT output power takes considerably less time. Now it only takes about 5 ms to level the example DUT, compared to 150 ms with the traditional method. Notice that the hardware-based method takes one more step to level than the traditional method, but it requires much less time overall. As shown in the following figure, the first several steps occur very quickly due to the shorter average time, then occur farther apart as the loop converges on a leveled state.

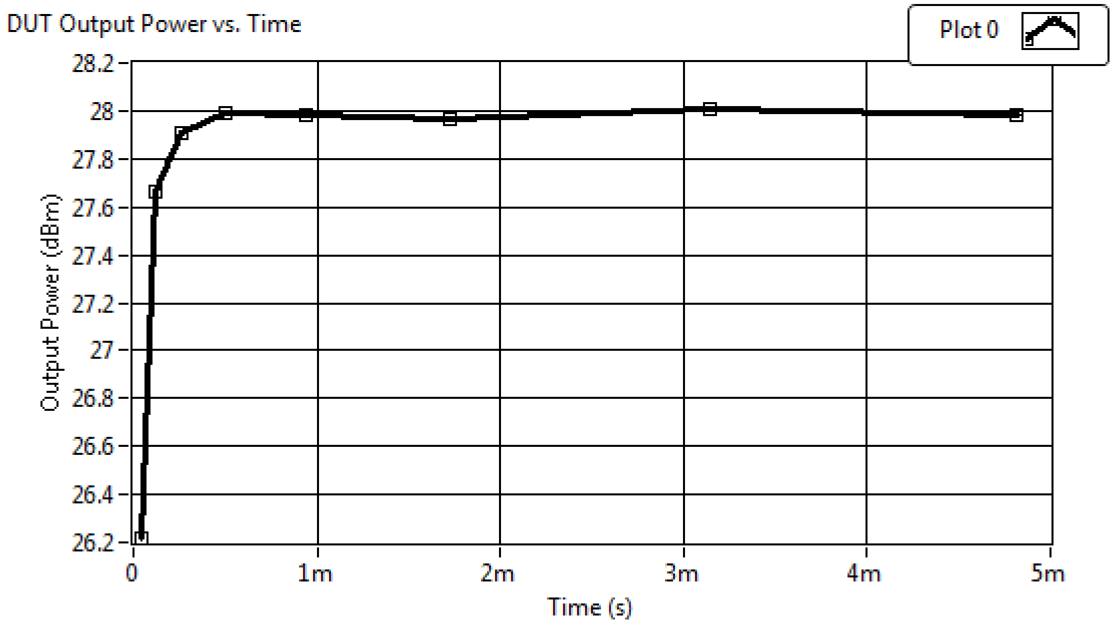


Figure 8. Graph of Hardware-Based Leveling

Limits for Determining the DUT is Leveled

The Auto Level IP uses limits to ensure leveling continues until the DUT reaches the desired output power level. Typically, you would consider the DUT output power successfully leveled when it is within the desired range, such as ±0.05 dB. However, if the output power is still changing quickly, the algorithm might stop before leveling completes. In the following figure, for example, while the first measured point is within the desired range, shown by red dashed lines, the second point is not. So, leveling should continue.

The Auto Level IP continues the leveling process until the following limits are satisfied:

- The averaging schedule reaches the minimum number of steps (for example, using the final averaging time).
- The measured output power is within the specified limits.
- The VSG output power changes very little between the last step and the current step.

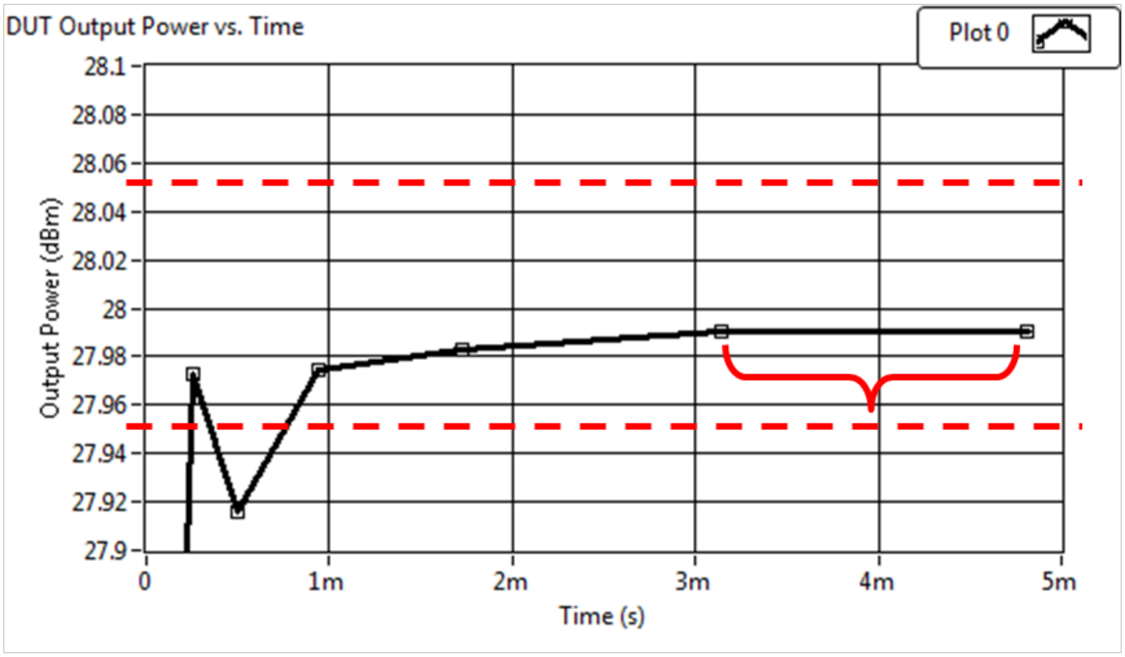


Figure 9. Limits for Determining the DUT is Leveled

Thermal Droop

A physical phenomenon associated with some PAs is that the gain continues to decrease for some time after initial leveling completes. This is known as *thermal droop*, or a *thermal tail*. The following figure shows this effect on the DUT used in this example under the traditional leveling method. The output of the DUT levels, and then the VSG output level, hold constant while the VSA continues to take measurements for another 10 s. Here you can see that the output levels to 28 dBm in about 150 ms. Then as the VSG output level holds constant the PA output power droops by 0.1 dB over the next 10 s. The entire thermal tail for the example DUT is in fact much longer than 10 s, and it continues to droop slowly for several minutes.

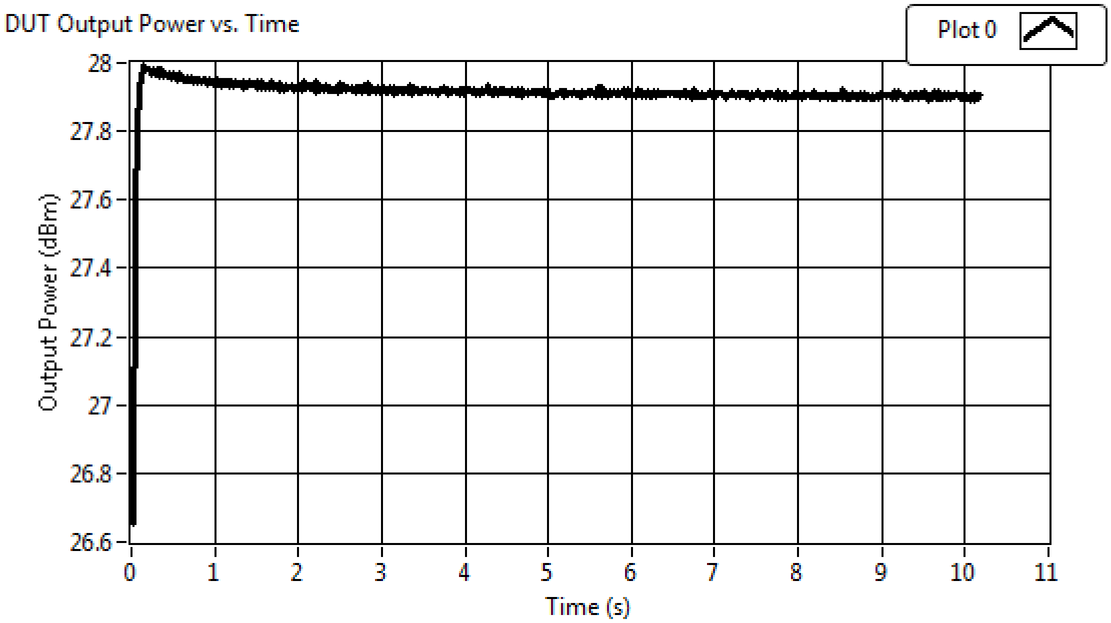


Figure 10. Thermal Droop with Traditional Leveling

After the DUT output power initially levels, you need to take performance measurements using the VSA. However, the thermal droop of the DUT output power affects the performance measurements. The traditional method of dealing with this effect slows down the leveling process to account for the *steep slope* part of the droop, which occurs in the first 2 s for the DUT in this example. This does help, but takes 2 s and does not fully account for the droop.

Rerunning the same experiment using the hardware-based leveling method, you see a little bit more droop. This droop occurs because the leveling completes much faster, so less of the *steep slope* part of the droop happens during the leveling loop.

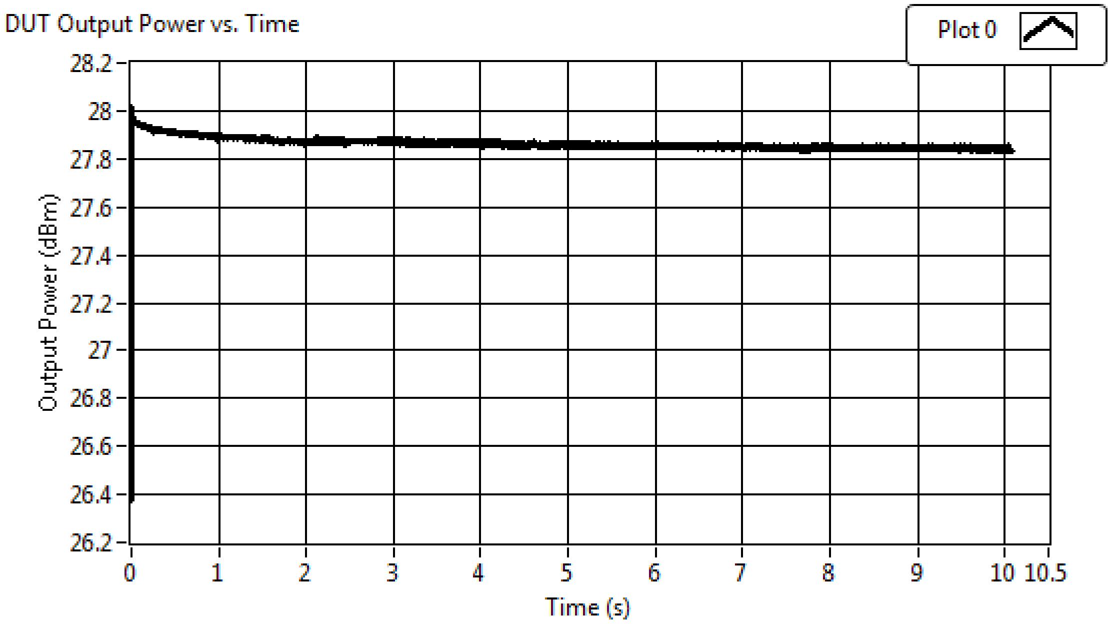


Figure 11. Thermal Droop with Hardware-Based Leveling

You can artificially slow down the hardware-based leveling method by setting a minimum number of iterations required before leveling completes. You now have the option to continue leveling during the *steep slope* part of the droop for DUTs with a lot of droop, and the option to level quickly for DUTs with only a little droop. These options are an improvement over the traditional method. However, there is a better method.

The hardware-based method uses a VSA to take both performance measurements and power level measurements while the Auto Level IP runs in parallel to the rest of the VSA and VSG data paths. So, you can continue making minor adjustments to the VSG output power level while measuring performance after the initial leveling completes.

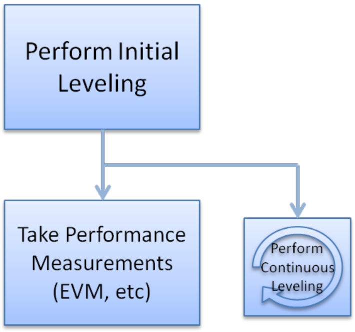

Perform Initial Leveling
Take Performance Measurements (EVM, etc)
Perform Continuous Leveling

Figure 12. Continuous Leveling

When you enable continuous hardware-based leveling, you get the results shown in the following figure. In this case the initial leveling occurs in about 5 ms, and then the VSG output roughly holds constant for the next 10 s. The VSG output updates in approximately 0.001 dB steps to compensate for the droop in the DUT output. The DUT output power level now holds constant. Because the continuous leveling occurs in parallel to the normal VSA data path, you can take performance measurements at the same time.

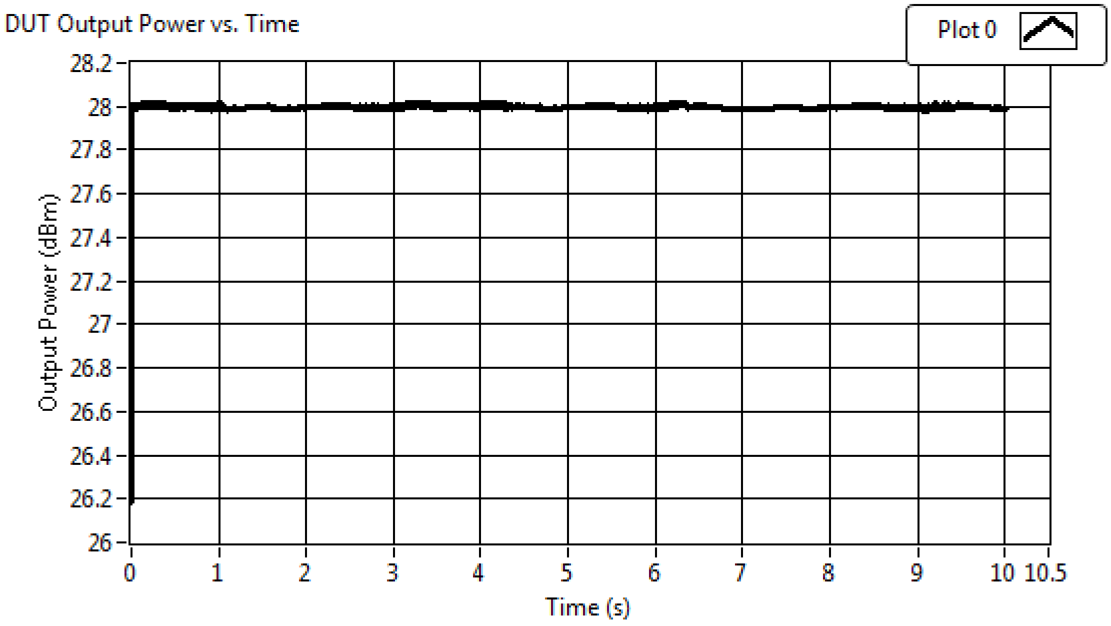
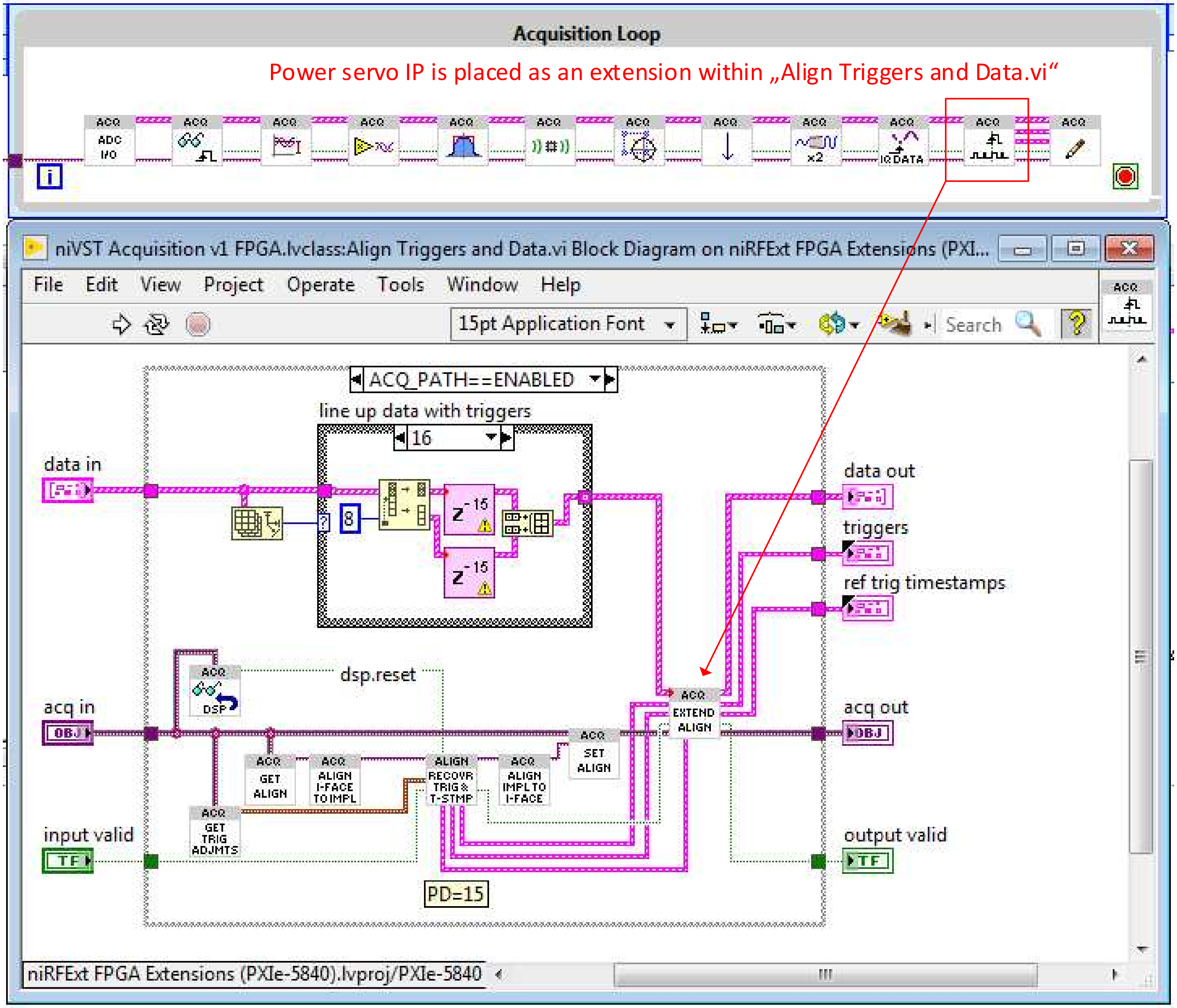


Figure 13. Continuous Hardware-Based Leveling

FPGA Details

The Power Servoing IP is an FPGA extension that you place in the digital signal processing (DSP) chain of the VSA I/Q data path. Include it immediately before Write Waveform, which transfers the I/Q data to memory. During the initial power leveling, Power Servo only consumes the I/Q data and does not transfer it to memory. After the leveling process completes, the data forwards to memory with Write Waveform.



Power servo IP is placed as an extension within „Align Triggers and Data.vi“

Figure 14. VSA Data Path

The result of power servoing is a new digital gain value, which is applied inside the interpolation extension in the VSG I/Q data path (see the following figure). If you set that digital gain value to `1`, the VSG outputs as normal. You can also reduce the digital gain using the Power Servoing IP to linearly control the VSG output level.

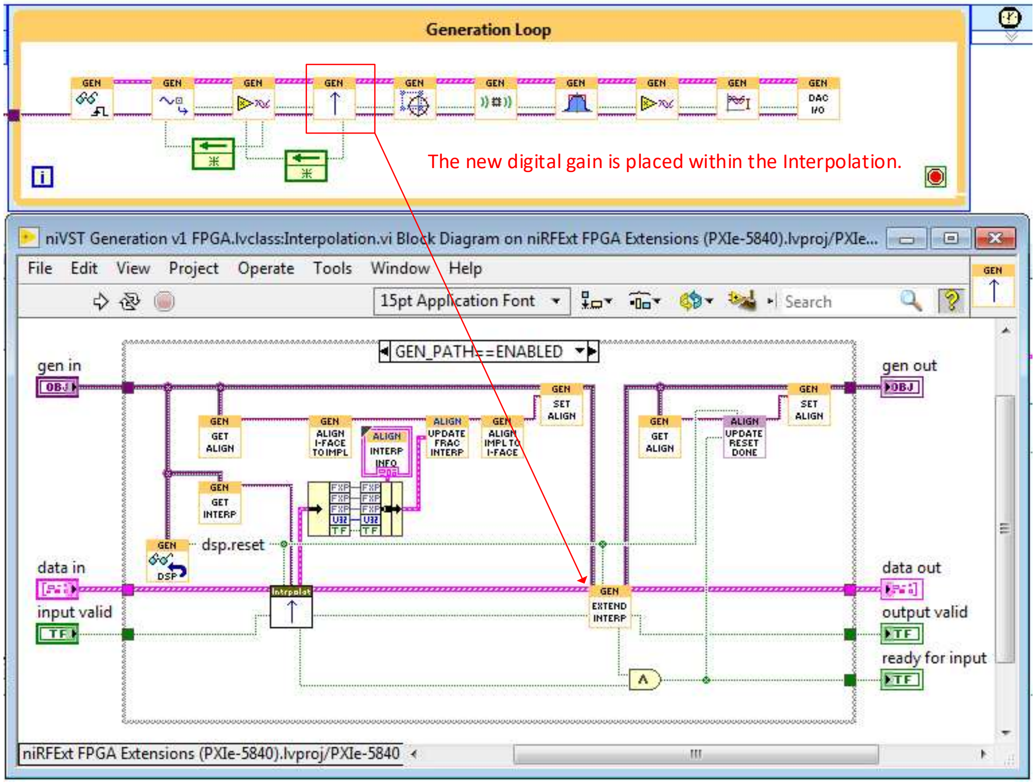

The new digital gain is placed within the Interpolation.

Figure 15. VSG Data Path with New Digital Gain

Auto Level.vi

A state machine that sequences through the leveling process. On the left there is a block RAM that contains the averaging schedule, and on the right are three VIs that accumulate the data, calculate the new gain value, and determine if the PA output is in range. The VI at the top configures the reference trigger based on whether servoing is complete and the burst mode counter has valid data.

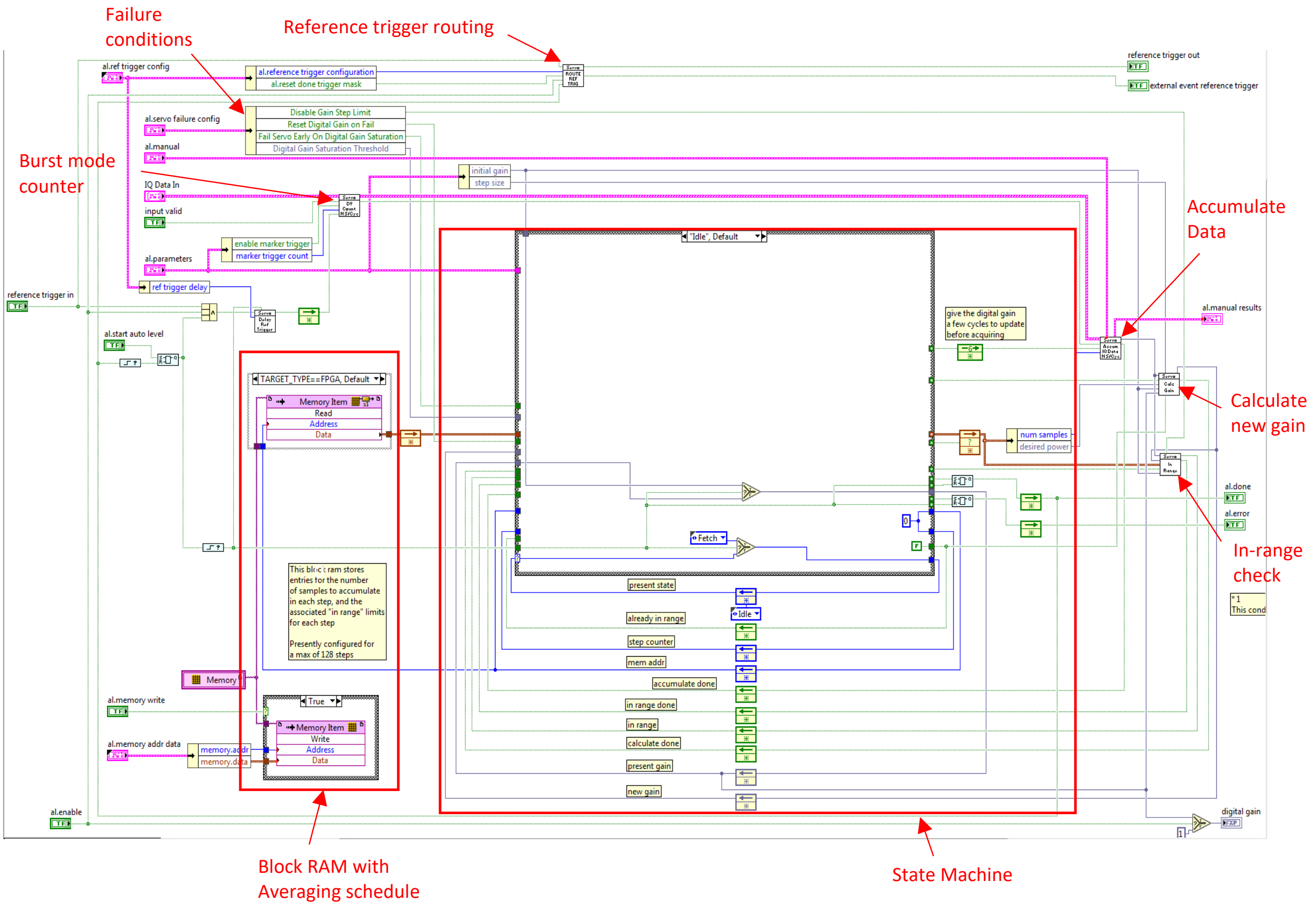


Figure 16. Auto Level VI Block Diagram

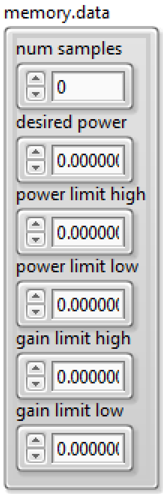
The state machine operates in the following order:

1. Waits for start.

   Note: Step 1 fetches a new entry from the block RAM each iteration until reaching the last entry, which it fetches for each remaining iteration.
2. Fetches the next averaging schedule entry from the block RAM.
3. Starts accumulating data and waits until done.
4. Starts calculating gain and waits until done.
5. Checks if the calculated gain is in range and proceeds according to the following table.

Gain in range	**Continuous mode?**	
	True	Updates the gain with the step size and returns to step 2.
	False	Sets **Done** to True and returns to step 1.
Gain not in range	**Reached maximum iterations?**	
	True	Sets the digital gain to the default value (if **Reset Digital Gain on Fail** is True), returns an error, and returns to step 1.
	False	Updates the digital gain (if it is not saturated) and returns to step 2.

The averaging schedule stored in the block RAM contains the parameters in the following figure.



memory.data

num samples: 0

desired power: 0.000000

power limit high: 0.000000

power limit low: 0.000000

gain limit high: 0.000000

gain limit low: 0.000000

Figure 17. Averaging Schedule Parameters

The **num samples** value sets the length of the accumulation for this step. Since the FPGA accumulates data rather than averaging it to avoid having to divide by the number of samples in the FPGA, the other parameters (**desired power**, **power limit high**, and **power limit low**) are a function of the number of samples and must be stored with the number of samples in the block RAM.

Accumulate IQ Data.vi

Calculates $|ADC\ Data|^2$ and accumulates **num samples** in a row. Use the **al.manual** control to take a manual measurement when the state machine is not running.

Note: Only valid samples are counted and accumulated.

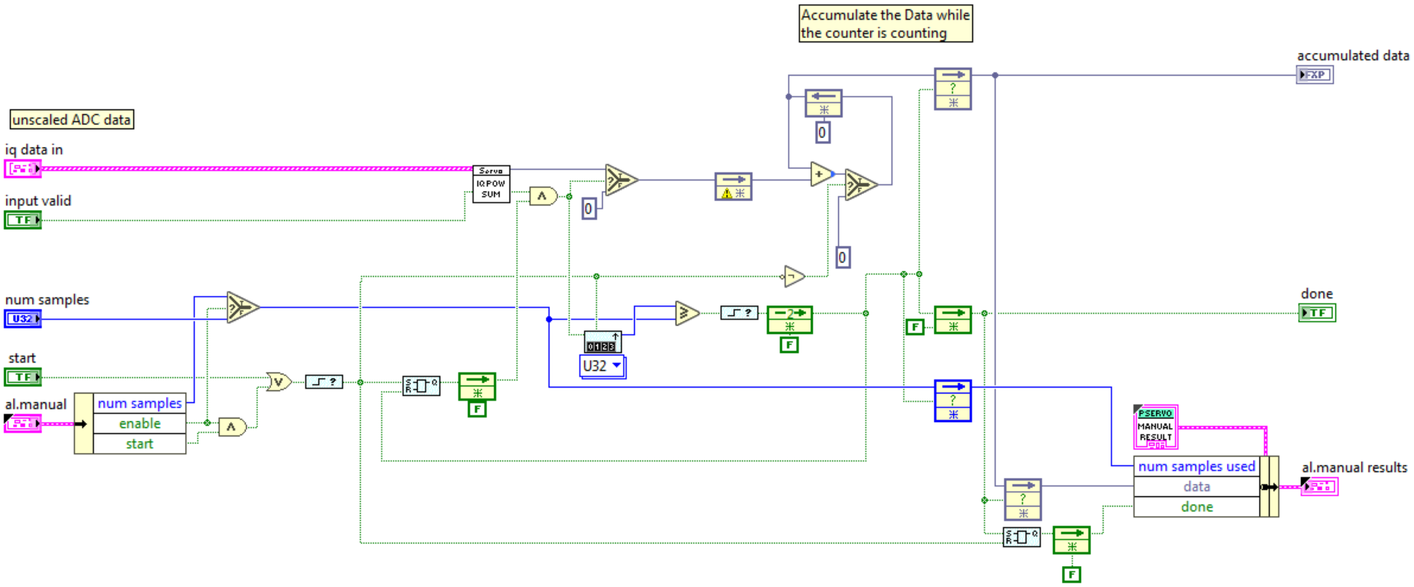


Figure 18. Accumulate IQ Data Block Diagram

Calculate Gain.vi

Non-Continuous Mode

Divides the desired power by the accumulated power to create a new digital gain value. Use non-continuous mode while the DUT is leveling. Refer to the Calculation Details section mathematical information.

Note: The divide, square root, and multiply operators take many cycles per sample, since the VI only calculates one digital gain value for many accumulated samples. These cycles greatly reduce resource usage and compile time with minimal effect on the overall leveling time.

Continuous Mode

Adds or subtracts the step size of small adjustments to the digital gain. Use continuous mode to continue making very small adjustments to the digital gain after leveling completes.

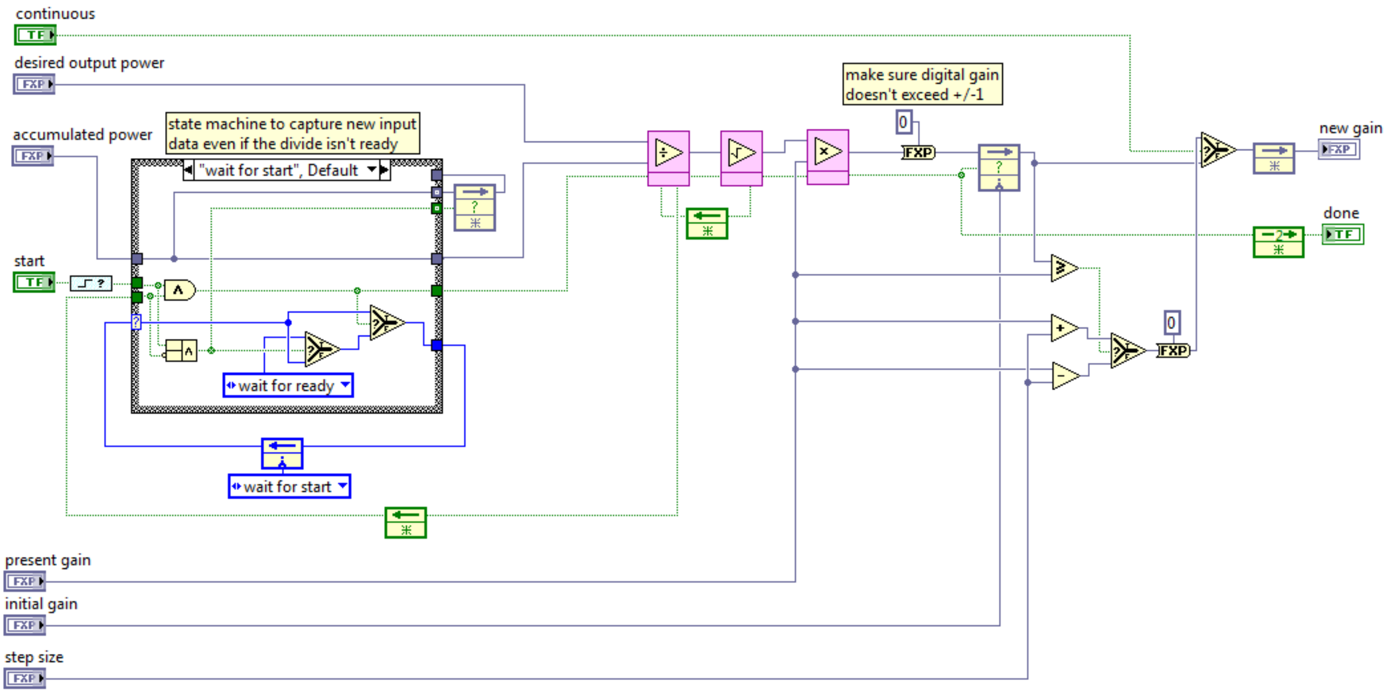

continuous
desired output power
accumulated power
state machine to capture new input data even if the divide isn't ready
"wait for start", Default
start
wait for ready
wait for start
make sure digital gain doesn't exceed +/-1
0
FXP
new gain
done
present gain
initial gain
step size

Figure 19. Calculate Gain Block Diagram

In Range.vi

Checks if the accumulated data is within specified power limits. When the **Disable Gain Step Limit** condition is False, checks if the change in digital gain is small (within the gain limits) to prevent stopping too early. Refer to the Limits for Determining the DUT is Leveled section for more information on stopping the leveling process.

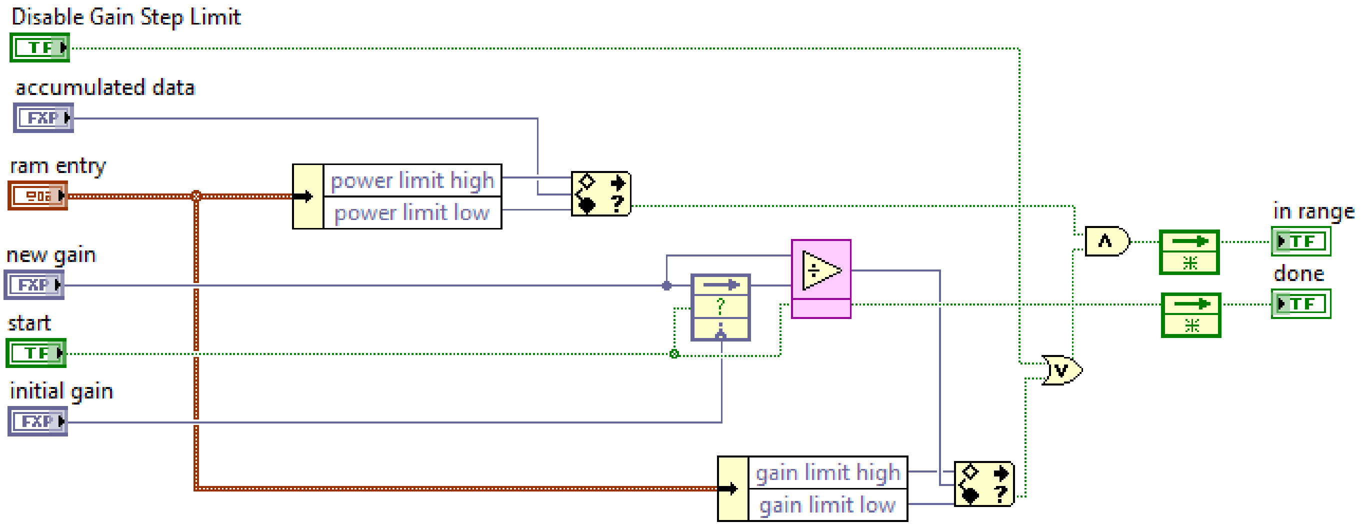
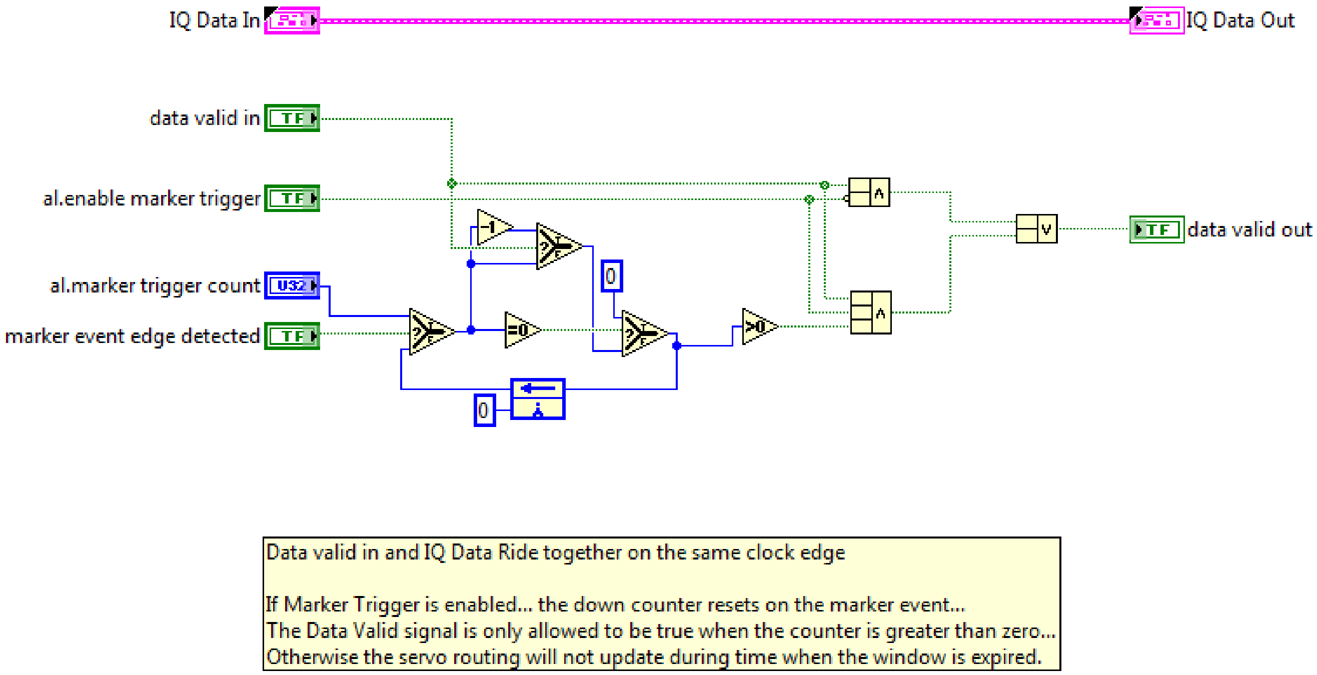


Figure 20. In Range Block Diagram

Data Valid Counter (Burst Mode Servo).vi

Detects a marker event and only allows the number of samples specified in **al.marker trigger count** to pass through. Disable the marker trigger to pass all valid data through.



Data valid in and IQ Data Ride together on the same clock edge

If Marker Trigger is enabled... the down counter resets on the marker event...
The Data Valid signal is only allowed to be true when the counter is greater than zero...
Otherwise the servo routing will not update during time when the window is expired.

Figure 21. Data Valid Counter (Burst Mode Servo) Block Diagram

Route Reference Trigger.vi

Drives the following configurations for the reference trigger:

- Asserts immediately when the servo completes
- Asserts as specified after the servo completes

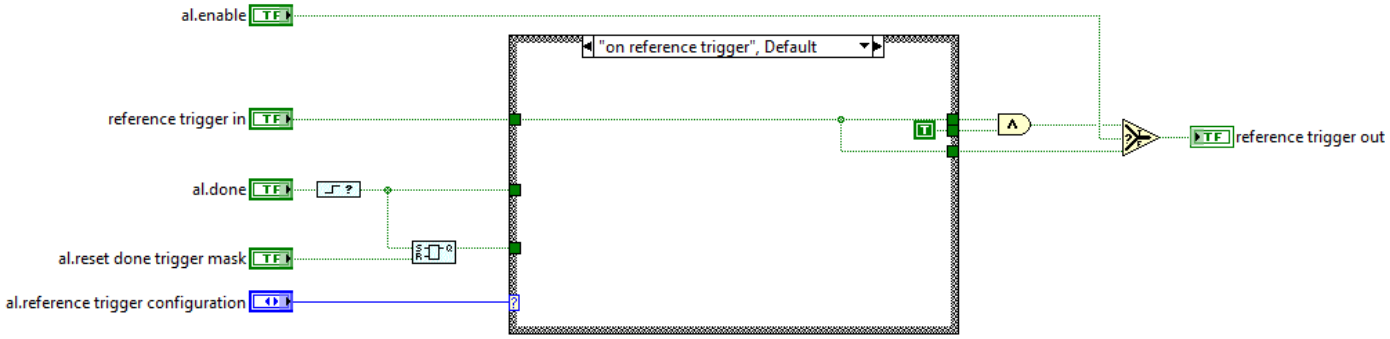

al.enable
reference trigger in
al.done
al.reset done trigger mask
al.reference trigger configuration
"on reference trigger", Default
reference trigger out
TF

Figure 22. Route Reference Trigger Block Diagram

Calculation Details

The power, voltage, and units need to match the data that accumulates.

RF In Configure returns the *Downconverter Gain* value, which represents how to scale the raw analog-to-digital converter (ADC) data to volts.

Convert volts to instantaneous power (in watts) using the following equation:

$$P = \frac{V^2}{R}$$

$$P = \frac{|A * S|^2}{2 * 50}$$

where

A represents raw ADC Data

S represents the scaling factor

$Scaling\ Factor = 10^{(\frac{-(Downconverter\ Gain\ (dB))}{20})}$

$Received\ Signal\ (Volts) = A * S$

Note: There is an extra factor of 2 in the denominator because the ADC data is complex.

Compute the average power (in watts) with the following equation:

$$P_{ave} = \frac{1}{N} * \sum_{n=1}^{N} \frac{|A_n * S|^2}{100}$$

Convert to dBm using the following equation:

$$P_{ave} = 10 * \log(\frac{1000}{N} * \sum_{n=1}^{N} \frac{|A_n * S|^2}{100}) = 30 + 10 * \log(\frac{1}{N} * \sum_{n=1}^{N} \frac{|A_n * S|^2}{100})$$

Implementing the constant multiplications and divisions in the FPGA would be a waste of resources. To make the math in the FPGA as simple as possible, the FPGA accumulates $|A|^2$ instead. You must convert values in dBm to match that unit. In the following equations, P_{ave} refers to an average power parameter (such as desired power, high limit, low limit, and so on) that you need to translate into the unit of what accumulates in the FPGA.

$$P_{ave} = 10 * \log(\frac{1000}{N} * \sum_{n=1}^{N} \frac{|A_n * S|^2}{100})$$

$$\frac{10}{N} * \sum_{n=1}^{N} |A_n * S|^2 = 10^{\frac{P_{ave}}{10}}$$

$$\sum_{n=1}^{N} |A_n * S|^2 = \frac{N}{10} * 10^{\frac{P_{ave}}{10}}$$

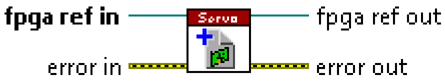
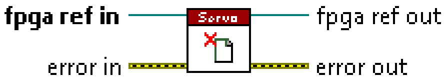
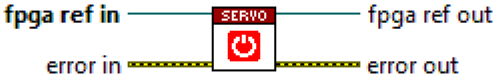
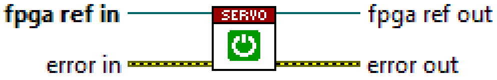
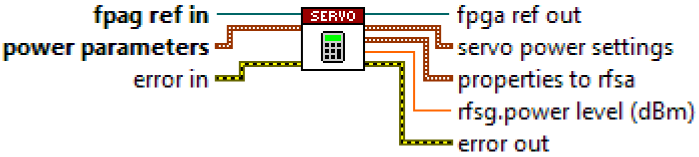
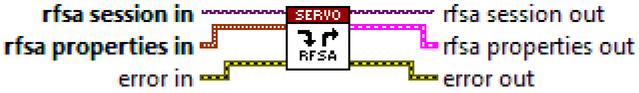
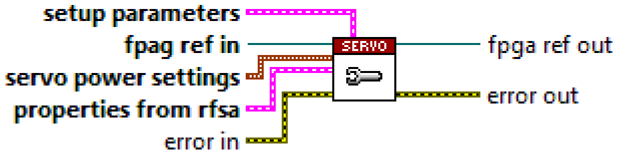
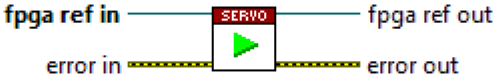
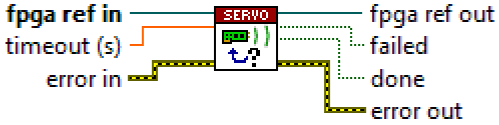
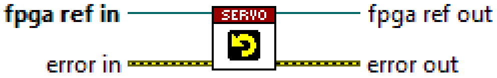
$$S^2 * \sum_{n=1}^{N} |A_n|^2 = \frac{N}{10} * 10^{\frac{P_{ave}}{10}}$$

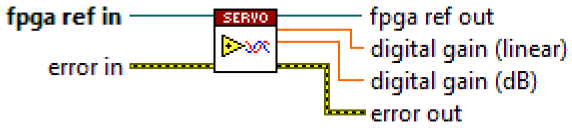
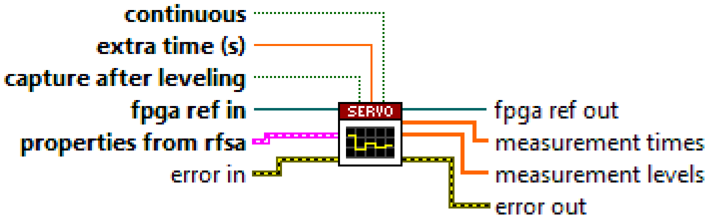
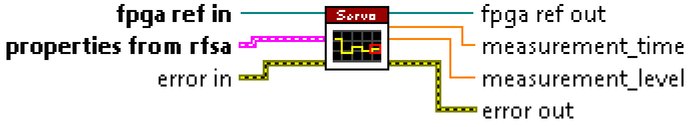
$$\sum_{n=1}^{N} |A_n|^2 = \frac{0.1 * N}{S^2} * 10^{\frac{P_{ave}}{10}}$$

$$\sum_{n=1}^{N} |A_n|^2 = \frac{N}{S^2} * 10^{\frac{P_{ave}}{10} - 1}$$

For example, if you accumulate 10,000 samples and want to set the high limit to 28 dBm, write $\frac{10000}{S^2} * 10^{\frac{28}{10}-1}$ to the FPGA as the high limit.

Host API

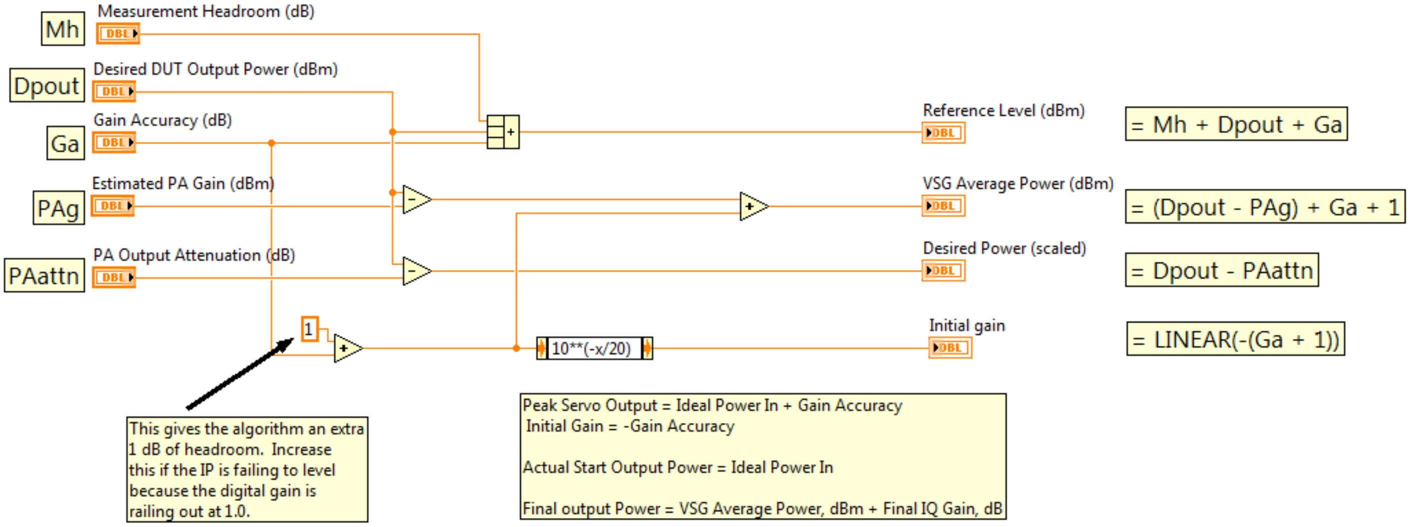
API State Handling	
Initialize.vi (Optional)  fpga ref in error in Servo fpga ref out error out	Creates a new API state handle linked to the FPGA reference.
Close.vi (Optional)  fpga ref in error in Servo fpga ref out error out	Closes an API state handle linked with the FPGA reference.
IP Enable	
IP Disable.vi  fpga ref in error in SERVO fpga ref out error out	Disables the FPGA Power Servo. Calling this function returns complete control of the RF output power level to the RFSG driver.
IP Enable.vi  fpga ref in error in SERVO fpga ref out error out	Enables the FPGA Power Servo. **Note**: Calling this function permits the IP to modify the value of the output power beyond the scope of RFSG.
Configuration	
Calculate Servo Parameters.vi  fpag ref in power parameters error in SERVO fpga ref out servo power settings properties to rfsa rfsg.power level (dBm) error out	Calculates power settings for RFSA, RFSG, and FPGA Power Servo.
Set & Get RFSA Properties.vi  rfsa session in rfsa properties in error in SERVO RFSA rfsa session out rfsa properties out error out	Sets and gets RFSA properties required for the FPGA Power Servo.
Setup.vi  setup parameters fpag ref in servo power settings properties from rfsa error in SERVO fpga ref out error out	Configures the FPGA Power Servo. Call Start to begin the servo after calling this VI.
State Control	
Start.vi  fpga ref in error in SERVO fpga ref out error out	Starts Servo Operation after setup is complete
Wait Until Done.vi  fpga ref in timeout (s) error in SERVO fpga ref out failed done error out	Blocks execution until the DUT output power has settled to the desired power within the desired accuracy level or until the specified timeout has been reached. Call after Start.
Reset.vi  fpga ref in error in SERVO fpga ref out error out	Resets the FPGA Power Servo to a known state.

Reporting	
Get Digital Gain.vi  fpga ref in error in SERVO fpga ref out digital gain (linear) digital gain (dB) error out	Returns the FPGA Power Servo digital gain. This, in conjunction with the RFSG power, gives the true power level.
Get Servo Steps.vi  continuous extra time (s) capture after leveling fpga ref in properties from rfsa error in SERVO fpga ref out measurement times measurement levels error out	Fetches servo data once the servo is done, regardless of whether it has passed or failed.
Get Last Servo Step Result.vi  fpga ref in properties from rfsa error in Servo fpga ref out measurement_time measurement_level error out	Fetches only the final servo step data and not the whole servo trace once the servo is done, regardless of whether it has passed or failed.

Calculate Servo Parameters.vi

Takes in a variety of factors about the DUT and produces the VSA reference level, the VSG output power level, and some parameters for the FPGA IP.

Tip: You can put an attenuator on the DUT output connected to the VSA input to avoid overloading the VSA. Report the amount of attenuation to the **PA Output Attenuation** control.



This gives the algorithm an extra 1 dB of headroom. Increase this if the IP is failing to level because the digital gain is railing out at 1.0.

Peak Servo Output = Ideal Power In + Gain Accuracy
Initial Gain = -Gain Accuracy

Actual Start Output Power = Ideal Power In

Final output Power = VSG Average Power, dBm + Final IQ Gain, dB

Figure 23. Calculate Servo Parameters Block Diagram

Details

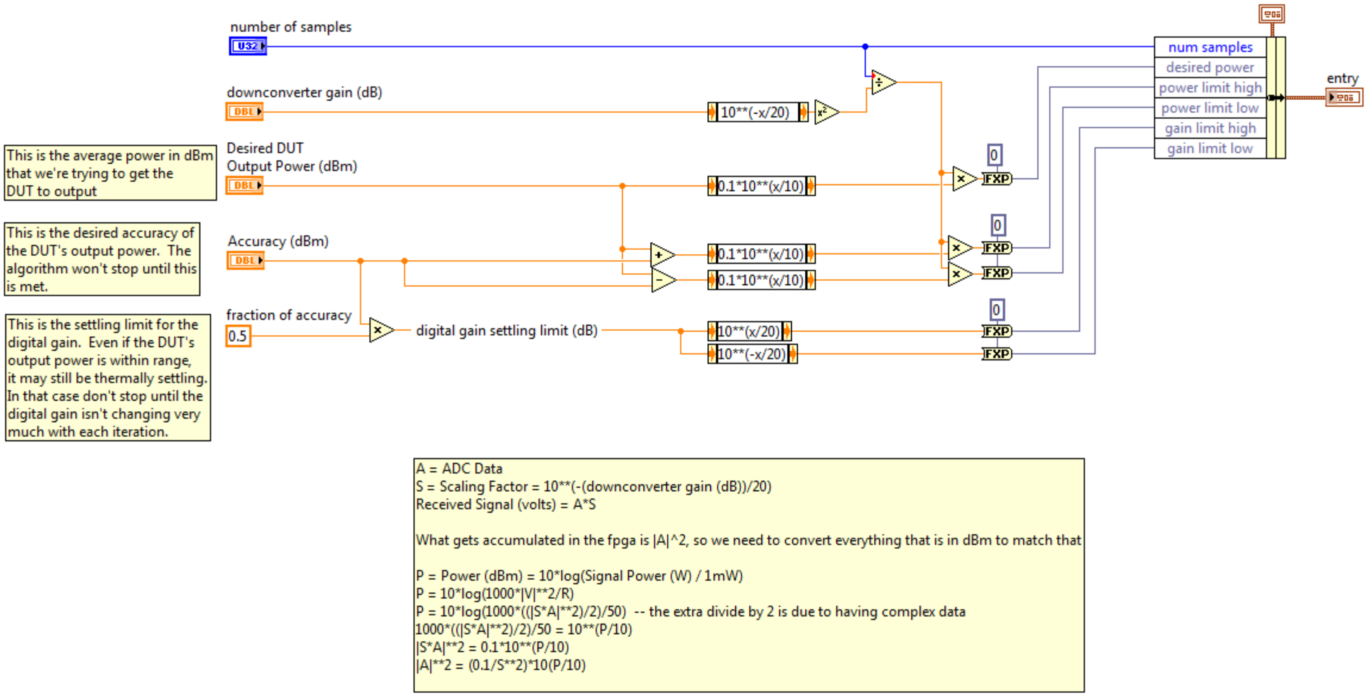
The *DUT Gain* is the typical gain specified in the data sheet. Use the gain accuracy to set the actual range over which the gain can vary. For example, if the data sheets lists gain as 26 to 30 dBm, set the **Estimated PA Gain** to 28, and the **Gain Accuracy** to 2.

The VI adds 3 dB of headroom to the VSA **Reference Level**. The default headroom was determined by experimentation. Adjust it as necessary for your application.

If you set the **Initial gain** too high, the digital gain maxes out at 1.0 before the power fully settles, and the Auto Level VI errors out. By default, the VI adds 1 dB of headroom to reduce the chance of failure. Increase the headroom if the Auto Level VI fails.

Build Averaging Schedule Entry.vi

Converts parameters from dBm to the unit of the accumulated data and creates an entry for the averaging schedule.



This is the average power in dBm that we're trying to get the DUT to output

This is the desired accuracy of the DUT's output power. The algorithm won't stop until this is met.

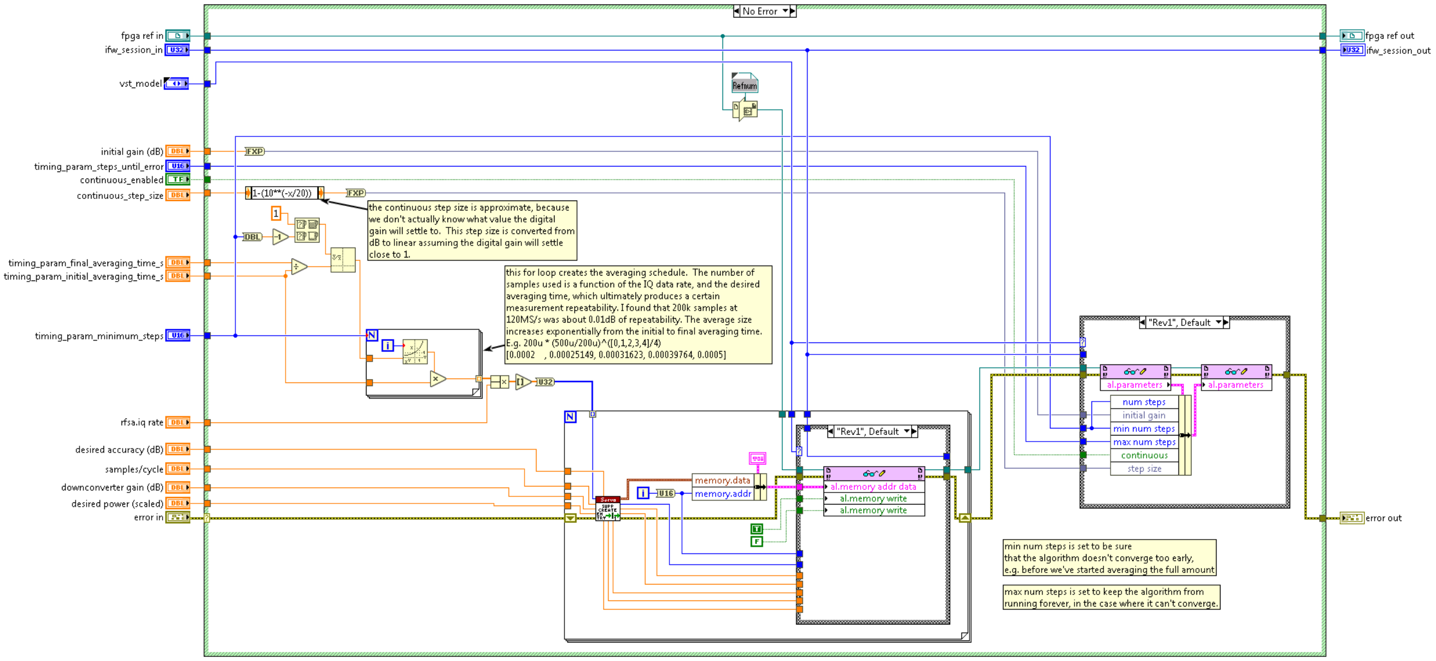
This is the settling limit for the digital gain. Even if the DUT's output power is within range, it may still be thermally settling. In that case don't stop until the digital gain isn't changing very much with each iteration.

```
A = ADC Data
S = Scaling Factor = 10**(-(downconverter gain (dB))/20)
Received Signal (volts) = A*S

What gets accumulated in the fpga is |A|^2, so we need to convert everything that is in dBm to match that

P = Power (dBm) = 10*log(Signal Power (W) / 1mW)
P = 10*log(1000*|V|**2/R)
P = 10*log(1000*((|S*A|**2)/2)/50)  -- the extra divide by 2 is due to having complex data
1000*((|S*A|**2)/2)/50 = 10**(P/10)
|S*A|**2 = 0.1*10**(P/10)
|A|**2 = (0.1/S**2)*10(P/10)
```

Figure 24. Build Averaging Schedule Entry Block Diagram

Calculate Configure Servo Schedule.vi

Creates the averaging schedule and forms the complete schedule and the parameters used to setup the Auto Level IP. Writes the calculated parameters to the FPGA. Use the **Averaging Schedule** cluster to choose the initial averaging time, final averaging time, and number of steps. The intermediate steps scale exponentially.



the continuous step size is approximate, because we don't actually know what value the digital gain will settle to. This step size is converted from dB to linear assuming the digital gain will settle close to 1.

this for loop creates the averaging schedule. The number of samples used is a function of the IQ data rate, and the desired averaging time, which ultimately produces a certain measurement repeatability. I found that 200k samples at 120MS/s was about 0.01dB of repeatability. The average size increases exponentially from the initial to final averaging time. E.g. 200u * (500u/200u)^([0,1,2,3,4]/4)
[0.0002 , 0.00025149, 0.00031623, 0.00039764, 0.0005]

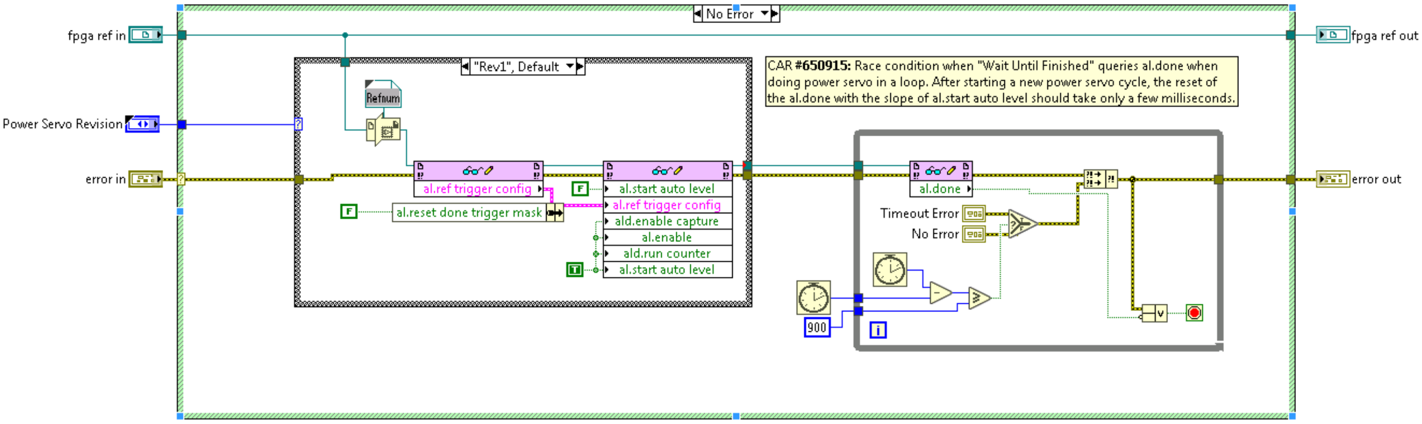
min num steps is set to be sure that the algorithm doesn't converge too early, e.g. before we've started averaging the full amount

max num steps is set to keep the algorithm from running forever, in the case where it can't converge.

Figure 25. Calculate Configure Servo Schedule Block Diagram

Start.vi

Sets **al.start auto level** to begin the leveling process. The revision input denotes either VST1 (**Rev1**) or VST2 (**Rev2**).



CAR #650915: Race condition when "Wait Until Finished" queries al.done when doing power servo in a loop. After starting a new power servo cycle, the reset of the al.done with the slope of al.start auto level should take only a few milliseconds.

Figure 26. Start Block Diagram

Check Servo Status.vi

Checks if the leveling process is complete. Leveling completes when **Done** is set and **error out** is not set.

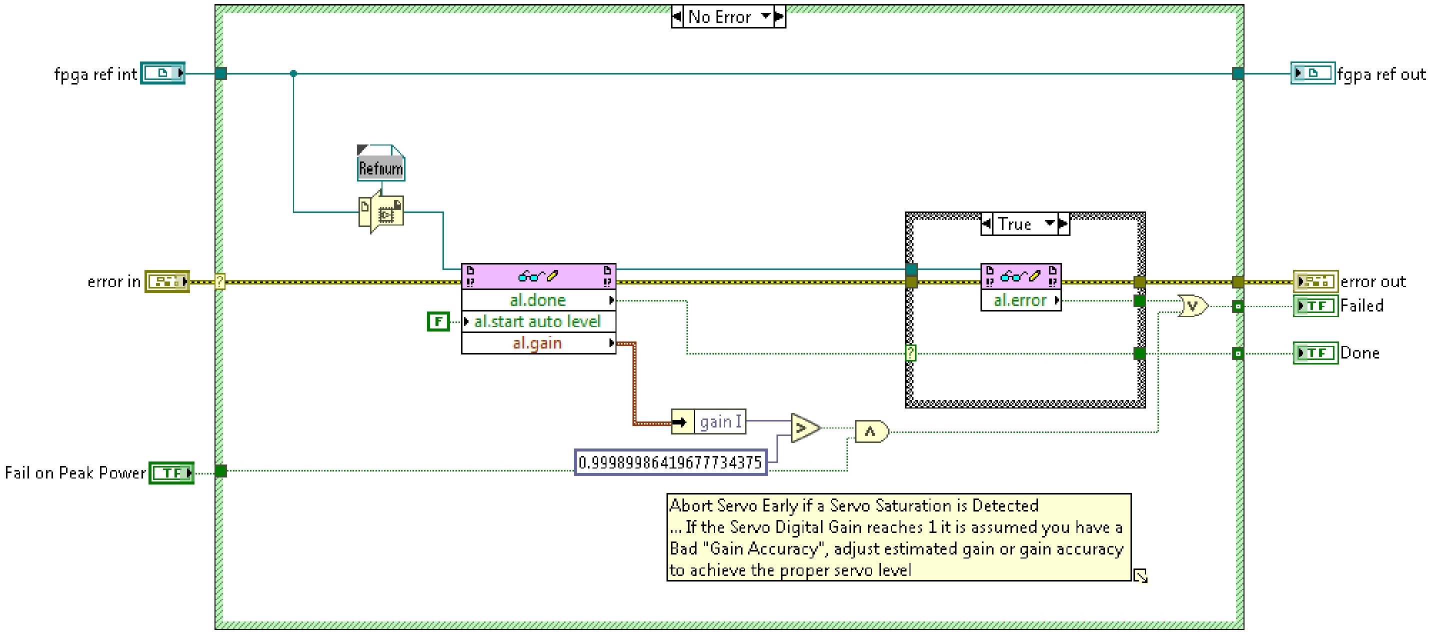

No Error
fpga ref int
fgpa ref out
Refnum
error in
al.done
F
al.start auto level
al.gain
True
al.error
error out
Failed
Done
gain I
0.99989986419677734375
Fail on Peak Power
Abort Servo Early if a Servo Saturation is Detected
... If the Servo Digital Gain reaches 1 it is assumed you have a
Bad "Gain Accuracy", adjust estimated gain or gain accuracy
to achieve the proper servo level

Figure 27. Check Servo Status Block Diagram

Reset.vi

Disables the Power Servoing IP if it is still running in continuous mode, resets the counters, and resets the servo done signal in Route Reference Trigger. If you do multiple servo operations, the burst mode configuration persists through all the succeeding servos. If you only do a single servo, the burst mode disables on reset.

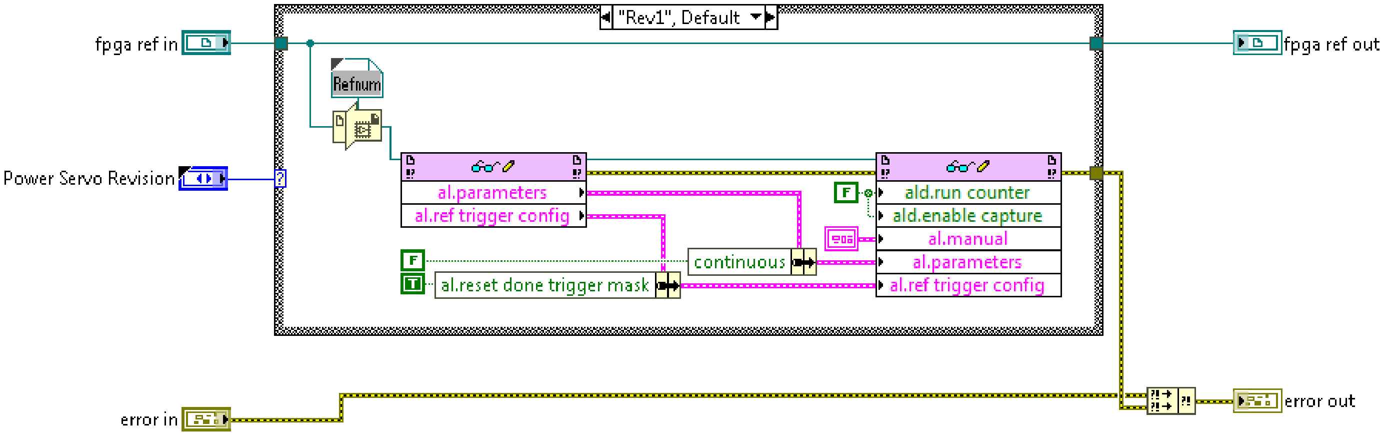


Figure 28. Reset Block Diagram

Get Servo Steps.vi

Captures extra data after servoing is complete and reads the power data captured during the servoing process. Converts the data to measurement times and levels.

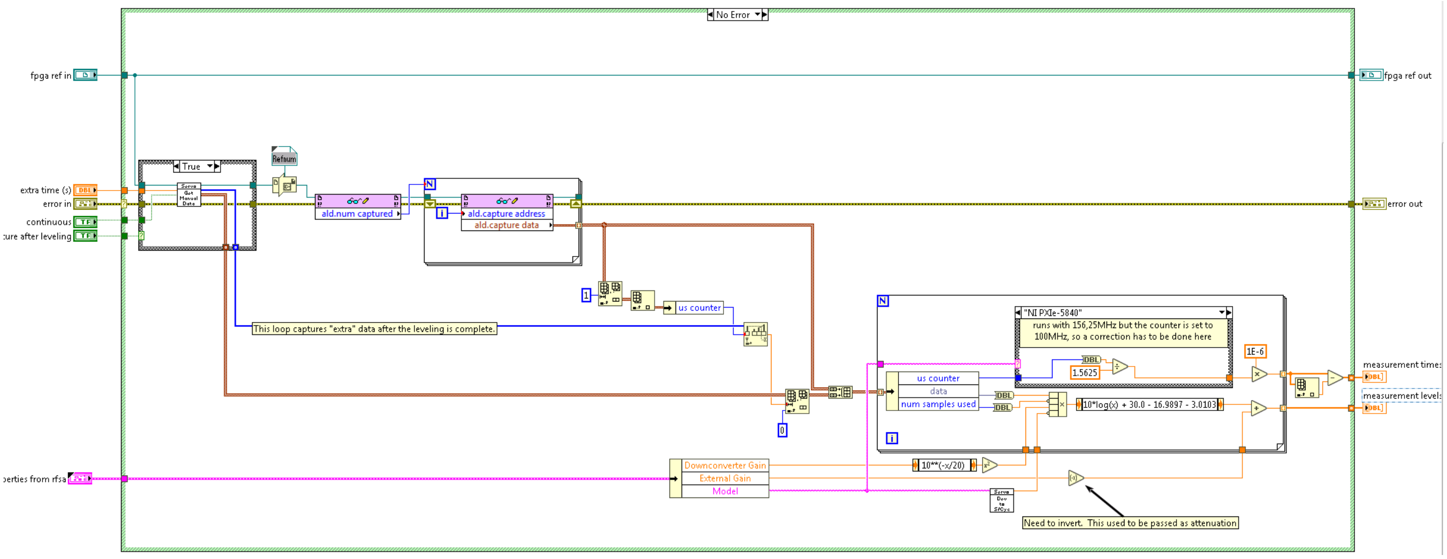


Figure 29. Get Servo Steps Block Diagram

Get Digital Gain.vi

Returns the raw digital gain from the FPGA and the digital gain in dB.

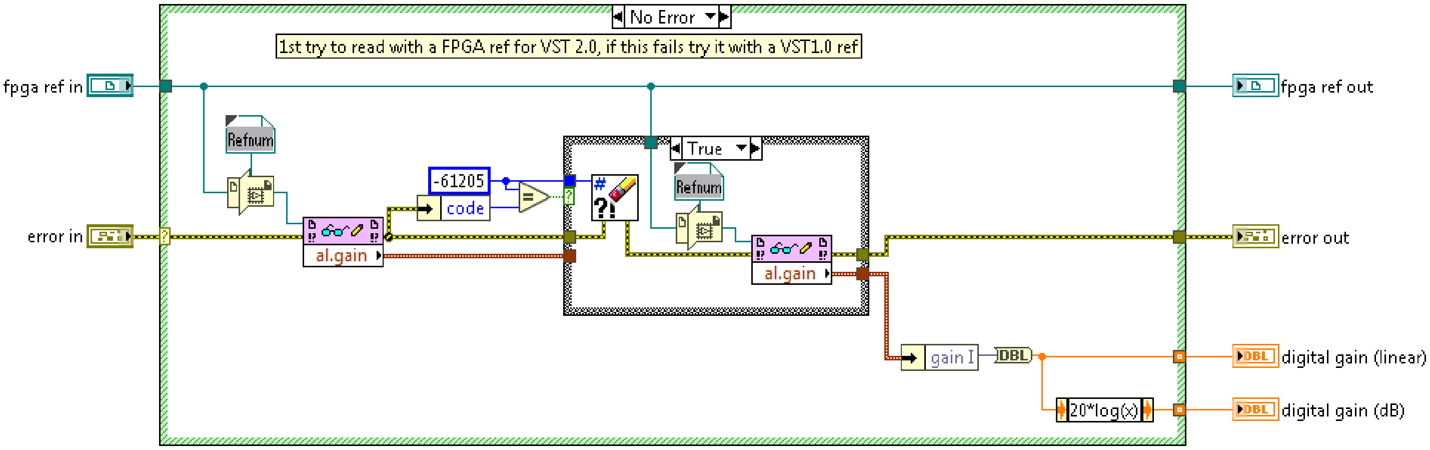


Figure 30. Get Digital Gain Block Diagram

Using the Power Servoing IP in LabVIEW

Complete the following steps to use the Power Servoing IP in LabVIEW.

1. (Optional) Create a new Power Servo session handle.
   Note: The initialize function is only available in RFIC Test Software 3.0 to current. In RFIC Test Software 3.0, the Power Servoing IP was moved into an extension of the NI-RFSA framework to be accessed through a unified interface. For backwards compatibility with the Power Servoing IP on VST1 (PXIe-5646 Vector Signal Transceiver), you can use your existing implementation.
2. Enable the Power Servoing IP and configure the following functions according to your use case.

   ### IP Enable.vi

   Enables the FPGA Power Servoing IP.

Calculate Servo Parameters.vi

Calculates the RFSG power level and power servo parameters (**initial gain**, **target output power**, and so on) based DUT related settings.

Setup.vi

Configures the Power Servoing IP with the following parameters.

- **Active Window Config**—Configures a time window within the I/Q samples used for power servoing. The internal RFSA reference trigger represents the start of the time window and the **active window length** parameter sets its duration. If the trigger is not at the beginning of the window, enter an additional delay with the **trigger delay** setting. The active window is typically used with test waveforms containing a signal burst and not used with continuous waveforms. If you start the power servo with a continuous waveform configuration, it immediately starts to accumulate provided I/Q samples.
  Note: The power servo only accumulates I/Q samples during the defined window length from the starting point. The time window signals when valid data samples can be used by the power servo block for the power calculation, which is independent from the averaging schedule. For example, the following waveform has a length of 200 usec with a 100 usec burst at the beginning. Assume the averaging schedule contains two servo steps with a duration of 150 usec per step. The averaging process requires three bursts, which totals 500 usec.

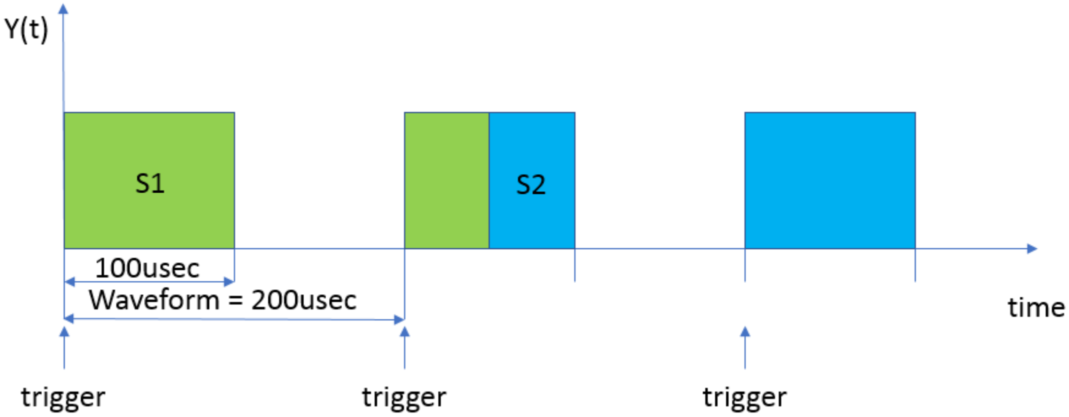


Figure 31. Graph of Burst Waveform with Triggers

If you start the power servo with a continuous waveform configuration, it immediately starts to accumulate provided I/Q samples.

You can only utilize the time window config in certain ways. The following corner cases demonstrate how different time adjustments are related.

Active Window Config Case 1

Use a continuous test waveform instead of a test waveform including a burst. Set the VSA acquisition time to match the waveform length. The VSG triggers the VSA to start the data acquisition.

In this scenario the VSA misses every second trigger from the VSG. Because there is a delay between signal generation (VSG) and data acquisition (VSA), a new trigger sends while the previous acquisition is still running. If the power servo processes more than one trigger, the overall power servo time increases by the waveform length. To prevent this, reduce the acquisition time as shown in the following figure.

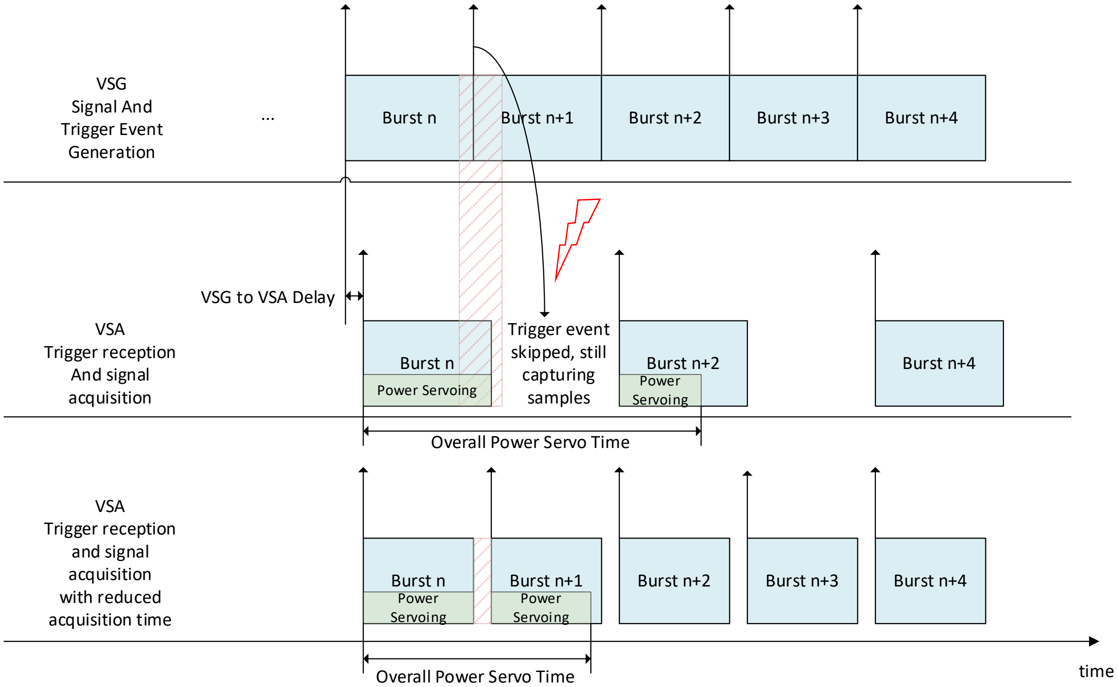


Figure 32. Active Window Config Case 1

Active Window Config Case 2

A single power servo step duration or the complete schedule time is the same length as the active window. The step time only includes the data acquisition time without the processing time needed by the power servoing block (the IQ sample accumulation, calculation of the digital gain, output power check and internal state transitions).

In this scenario, the power servo processing does not fit into a single active window. It needs an additional burst period, as shown in the following figure—two steps are configured with the same length as the active window. The processing time for PXIe-5840 was determined around 0.7 usec.

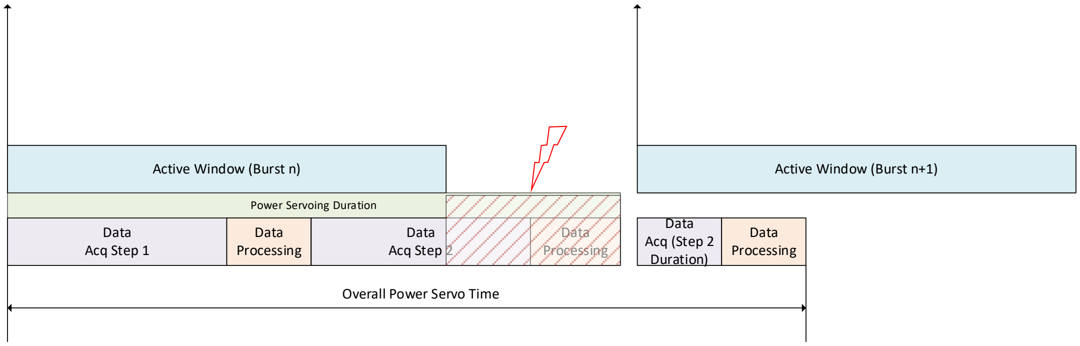


Figure 33. Active Window Config Case 2

- **VSA Reference Trigger Override Mode**—Gates the internal reference trigger in the Power Servoing IP and absorbs it during averaging. After Power Servo finishes, the VI either immediately generates a single reference trigger event or opens the gate to pass reference trigger events forward.
- **Create and Write the Averaging Schedule**
- **Configure Failure Condition**—Sets a saturation threshold for digital gain. The VI errors out if gain exceeds the threshold before executing the minimum number of steps

3. The Power Servoing IP executes the following steps:
    a. **Start.vi**—Starts the power servo.
    Note: You must have a valid signal acquisition running for power servoing to work. Therefore, start the signal generator (VSG) and the analyzer (VSA, here using RFmx SpecAn) before the power servo. The VSG generates the test waveform, which goes into the DUT. The analyzer acquires and measures the signal from the DUT. The power servo automatically levels the DUT output signal to the requested power based on your VSA settings (bandwidth, sample rate, and so on).
    b. **Wait Until Done.vi**—Blocks the execution until the DUT output power has settled to the desired power within the desired accuracy level or until the specified timeout has been reached.
    c. **Get Digital Gain.vi**—Fetches the power servo digital gain. The digital gain, in conjunction with the RFSG power, gives the true power level.
    d. **Get Servo Steps.vi**—Fetches the power servo trace showing the auto-level process.
    e. Execute the measurement based on the adjusted power level.
    f. Stop the generator.
    g. **IP Disable.vi**—Returns complete control of the RF output power level to the NI-RFSG driver. (This VI resets the RFSG digital gain back to unity.)
4. (Optional) If you created a Power Servo session handle, close it.
   Note: The close function is only available in RFIC Test Software 3.0 to current.

FPGA Power Servoing IP Specs

Resource Utilization of Xilinx Virtex-6 LX195T

Resource Percentage	**Utilization**
6.5%	8111 Look-Up Tables
4.0%	9895 Flip-Flops
4.4%	15 Block RAMs
0.3%	2 DSP Slices

The maximum achievable clock rate is 150 MHz with an empty FPGA.

Information is subject to change without notice. Refer to the *NI Trademarks and Logo Guidelines* at `ni.com/trademarks` for more information on NI trademarks. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering NI products/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patents Notice* at `ni.com/patents`. You can find information about end-user license agreements (EULAs) and third-party legal notices in the readme file for your NI product. Refer to the *Export Compliance Information* at `ni.com/legal/export-compliance` for the NI global trade compliance policy and how to obtain relevant HTS codes, ECCNs, and other import/export data. NI MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS. U.S. Government Customers: The data contained in this manual was developed at private expense and is subject to the applicable limited rights and restricted data rights as set forth in FAR 52.227-14, DFAR 252.227-7014, and DFAR 252.227-7015.

378117A-01 July 2019