

SYNTHESIS OF COMPLEX SIGNALS ON TEST EQUIPMENT

Matt Cornish, Racal Instruments Ltd, UK +44 01202 872800, matt.cornish@racalinst.co.uk
Chris Gorringe, Racal Instruments Ltd, UK, +44 01202 872800, chris.gorringe@racalinst.co.uk
Jim Langlois, RAF Wyton, UK

ABSTRACT

A traditional signal definition exhaustively lists every attribute and condition that it was thought the signal could possibly have. However, it is often the case that we are only interested in controlling a limited number of signal attributes and conditions. The traditional method is inflexible in this respect, leading to problems when trying to map different capabilities in a signal standard, as required for ATP (Automated Test Program) portability.

In this paper we will consider a concept called a Signal Graph to define a model of a single, complete signal. Each component in the Signal Graph identifies a feature or behavior that our signal will exhibit. The Signal Graph is a mechanism whereby we can define a set of features that we require, which can then be mapped onto a series or collection of suitable resources (e.g. algorithm, PC card, bus instrument, etc.).

We will show how complex, 'user defined' signals can be created and then mapped onto real hardware to provide a portable ATE (Automatic Test Environment) signal description, which can be used on a variety of ATEs.

This paper draws upon IEEE standard SDTD (Signal Definition and Test Description) [1] standard to utilize common components and interfacing techniques and shows how a single test description can be used for both simulation and execution.

Keywords. ATE, ATP, signal, portability, portable, synthesis, A2k.

1 INTRODUCTION

Traditionally, we have used very broad definitions of signals, listing every attribute and condition that the signal could possibly have. For example, an ATLAS [2] definition of a sinusoid must include some thirty parameters (used to define any AC signal), whereas, we are concerned with, perhaps, only three of these parameters. This, 'one model fits all', solution causes problems when trying to map the capabilities of a signal standard to particular instruments, creating large and difficult-to-maintain signal descriptions and hindering ATP portability.

Consider the idea of building these complex signals from the signal components that we actually use: -

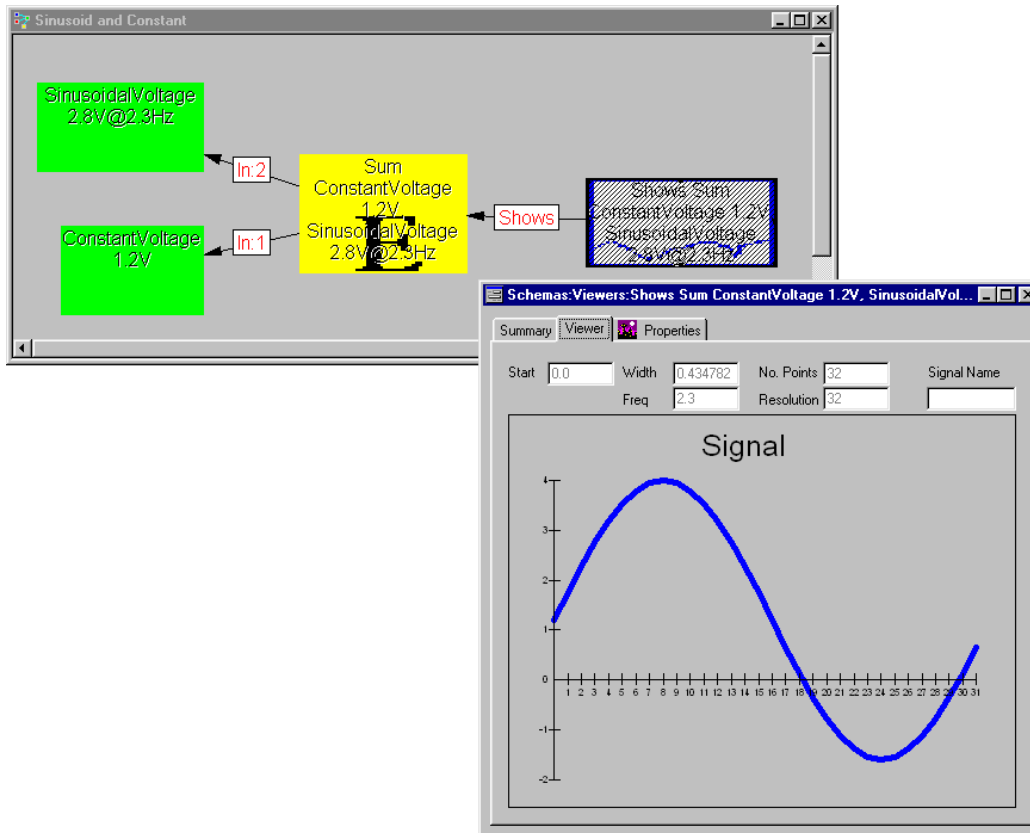


Figure 1. Example Signal Graph and Simulation

This concept, called a Signal Graph, models a complete signal, each component identifying a feature or behavior that our signal will exhibit. So, for our simple AC signal (Figure 1) we have the following components and attributes: -

SinusoidalVoltage component that conforms to the equation $A_0 \sin(\omega t + \theta)$

- amplitude of type Voltage, representing the A_0 value
- frequency of type Frequency, where ω represents $2\pi \times f$
- phase of type Radians, representing the θ value

ConstantVoltage component

- amplitude of type Voltage, representing the fixed DC offset.

Signal Graphs provide a mechanism with which we can define a set of features that we require, which can then be mapped onto a series or collection of suitable resources. Because we are only defining the features needed to describe our signal, it becomes possible to map to a larger number of possible candidates for our resources (e.g. algorithm, PC card, bus instrument, etc.)

There are a number of ways in which users can define signals using these signal components: One is by using native test programs to build Signal Graphs through an automation interface; Another is by defining a static signal, using database schema, through XML (Extensible Markup Language) [3] description. Both methods result in the same Signal Graph and, therefore, the same signal description.

Any resource allocation or mapping process becomes one of finding resources (Figure 2) that export the components of our Signal Graph (Figure 3) in the same configuration. This information could be obtained via some XML description, or by automatic interrogation of live resource objects. Once a set of suitable resource candidates has been identified (Figure 4), we can further refine the process by considering switching information and detailed attribute considerations, such as range, resolution etc.

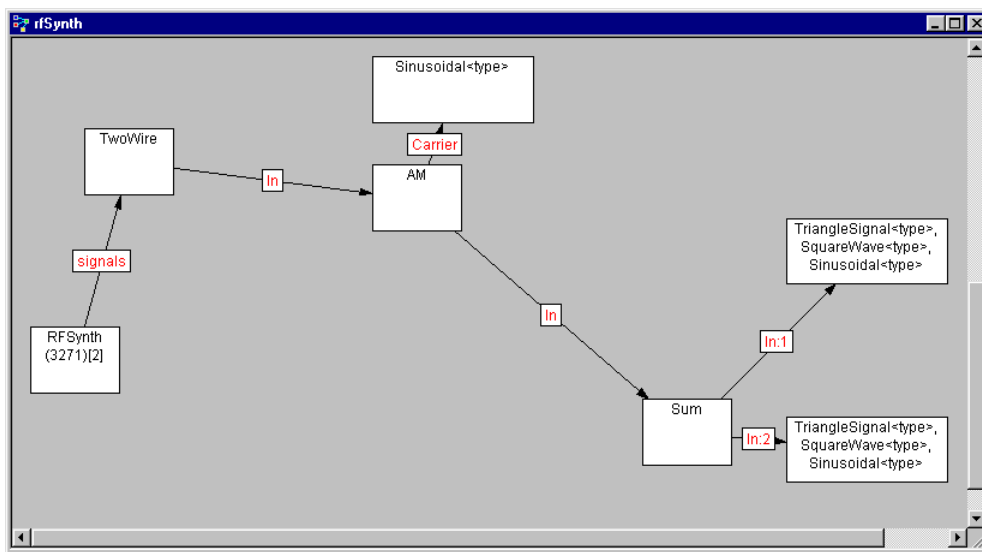


Figure 2. Resource Description for a Racal 3271 Signal Generator

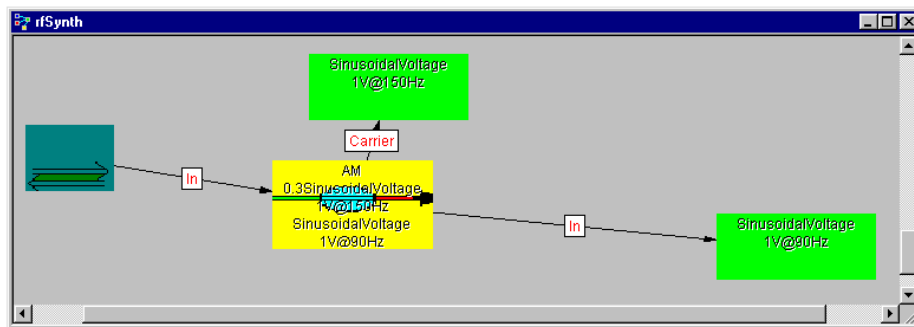


Figure 3. AM Signal Graph

By describing signals using these Signal Graph components, even complex signals can be mapped across ATEs and, also, transported to different ATEs. The ability to port signal

descriptions allows the carrier test program to be ported if the new ATE supports the same native language or XML parsers.

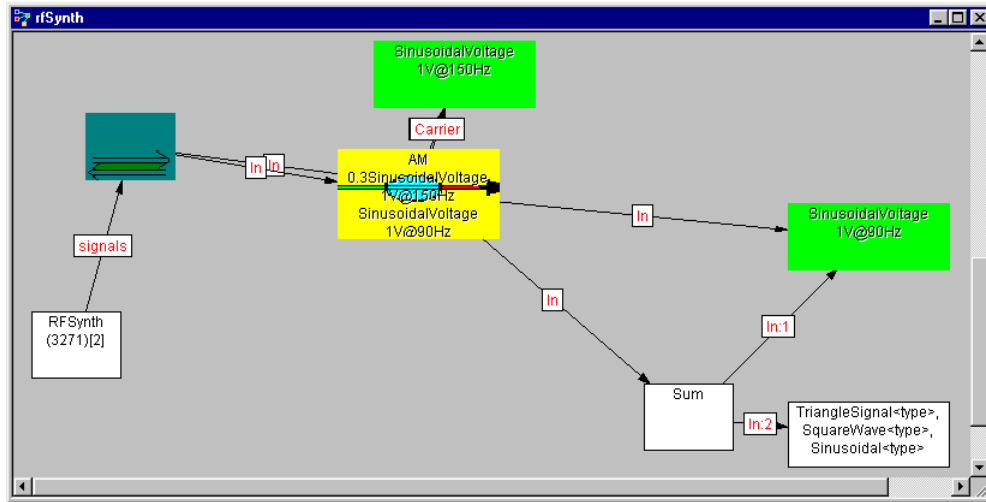


Figure 4. An Overlay, Showing the Mapping of an AM Signal Graph onto the Resource Description for a Racal 3271 Signal Generator

In order to achieve a common set of signal components we look to the IEEE SDTD standard. This standard defines common basic signal components and their interfacing techniques, as well as a set of pre-determined complex signals, which it defines in a TSF (Test Signal Framework) library. The set of TSFs currently covers a collection of components covering electrical (analogue & digital), distance, events and time together with connections. The standard also provides the names, properties and types of objects, the semantics of the signal features they describe and the rules necessary to allow multiple signals feature's to be combined using a Signal Graph. The standard provides enough information for the resulting signal descriptions to be simulated or executed on multiple ATE platforms.

2 MODELING

Using Signal Graphs, we are able to bring together different components to create complex signal descriptions (Figure 5).

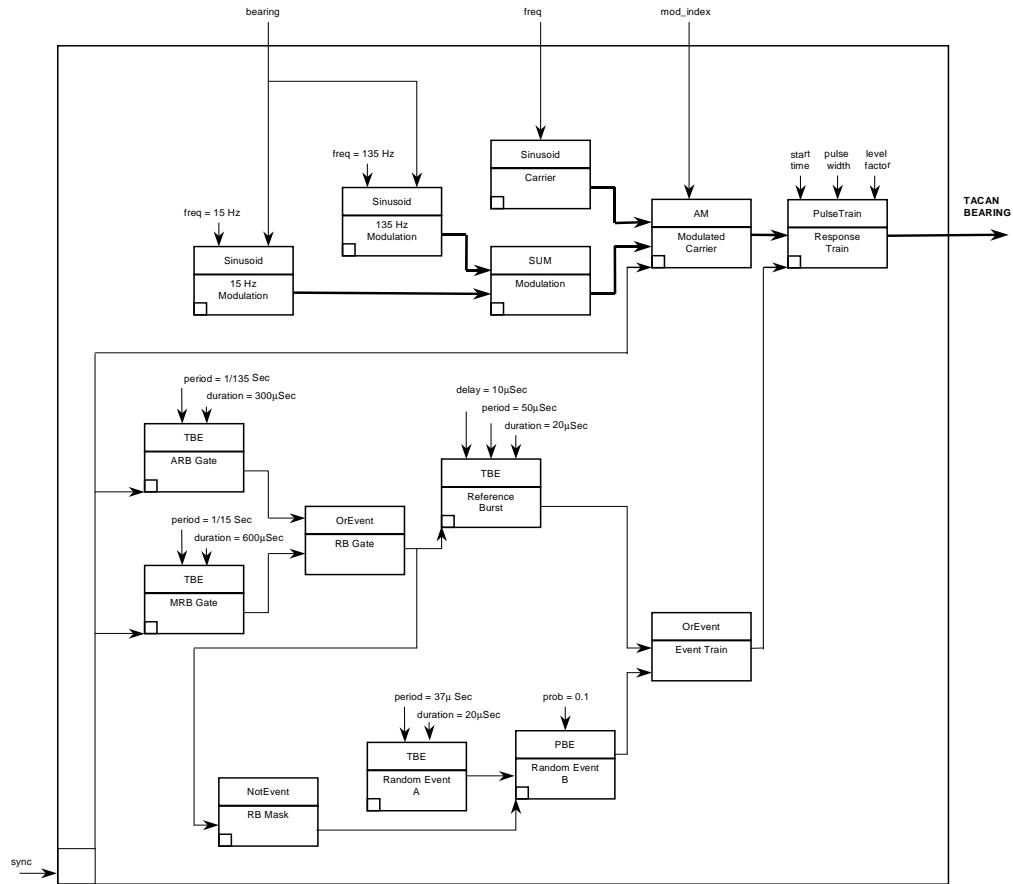


Figure 5. TACAN TSF

Another method of defining Signal Graphs is through computer programs, utilizing COM automation interfaces, allowing signals to be defined in any computer language. Such a test program should define and build the required signals. The signal is turned on with **Run** command and turned off with the **Stop** command. Note that here we are defining a signal and then controlling it, rather than worrying about controlling an instrument resource. So, for a simple sinusoid and DC offset, the Signal Graph would look something like that shown in Figure 6.

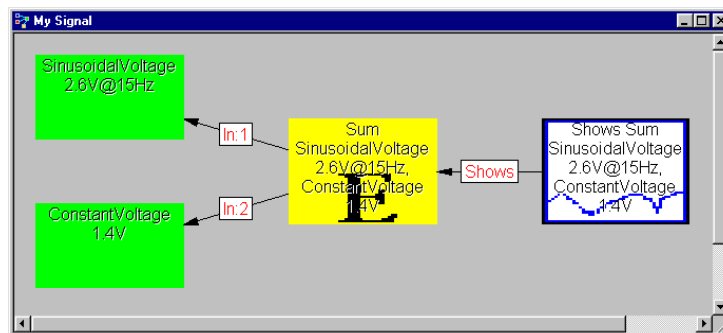


Figure 6. Example Signal Graph of a Sinusoid and DC Offset

An equivalent signal model to that shown in the Signal Graph (Figure 6) can be defined using VB (Visual Basic) [4] (Figure 7).

```
'Defines Sum ConstantVoltage 1.2V, SinusoidalVoltage 2.8V@2.3Hz

Dim A2k
Set A2k = CreateObject("Atlas2k.Resource")

Dim setup
Dim sSinAndConst(1)
'SETUP Constant amplitude 1.2 AS Constant8
Set Constant8 = A2k.Require("ConstantVoltage")
    Constant8.amplitude="1.2V"
Set setup = Constant8

'SETUP Sinusoidal amplitude 2.8, frequency 2.3 AS Sinusoidal17
Set Sinusoidal17 = A2k.Require("SinusoidalVoltage")
    Sinusoidal17.amplitude="2.8V"
    Sinusoidal17.frequency="2.3Hz"
Set setup = Sinusoidal17

'SETUP Sum AS Sum7
Set Sum7 = A2k.Require("Sum")
Set setup = Sum7
setup.In(1) = Constant8.Out
setup.In(2) = Sinusoidal17.Out

Set sSinAndConst(1) = Sum7
Sum7.Out.Run
```

Figure 7. Example VB Signal Definition of a Sinusoid and DC Offset

A further equivalent signal model can be defined using XML (Figure 8). Here we define a structured model using tag names, mapped on to the names defined within the standard for elements and attributes. Because the model we are defining is static, it does not change over time, though the signal the model describes most certainly does change over time. This allows us to define dynamic signals, using static models, without the need for explicit control statements. The XML represents a pure data capture and is an ideal medium of signal definition interchange.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- created with newWave (http://www.racalinst.com) by Chris Gorringe (Racal Instruments)
-->
<!--W3C Schema generated by newWave (http://www.racalinst.com) -->
<Signal name="SinAndConst" Out="Sum7"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="..\signals.xsd">
    <ConstantVoltage name="Constant8" amplitude="1.2V" />
    <SinusoidalVoltage name="Sinusoidal17" amplitude="2.8V" frequency="2.3Hz" />
    <Sum name="Sum7" In="Constant8 Sinusoidal17"/>
</Signal>
```

Figure 8. Example XML Signal Definition of a Sinusoid and DC Offset

All the above examples (Figures 6, 7 & 8) of signal definition are equivalent; the graphical IDE, the VB program and the XML description. In fact, we can write tools that interchange all of the above descriptions without affecting the signal definition.

There are no limitations to the complexity that these Signal Graph models can reach, as each component represents a separate behavior or modifies an existing component. Even multiple instances of components can be used with Signal Graphs.

2.1 Model Equivalents

Figure 9 shows a Signal Graph, modeling an ILS (Instrument Landing System) Localiser signal. Even if we cannot map this model directly on to our available resources, we may be able to find a model equivalent that does match our available resources. Figure 10 shows just such a model equivalent.

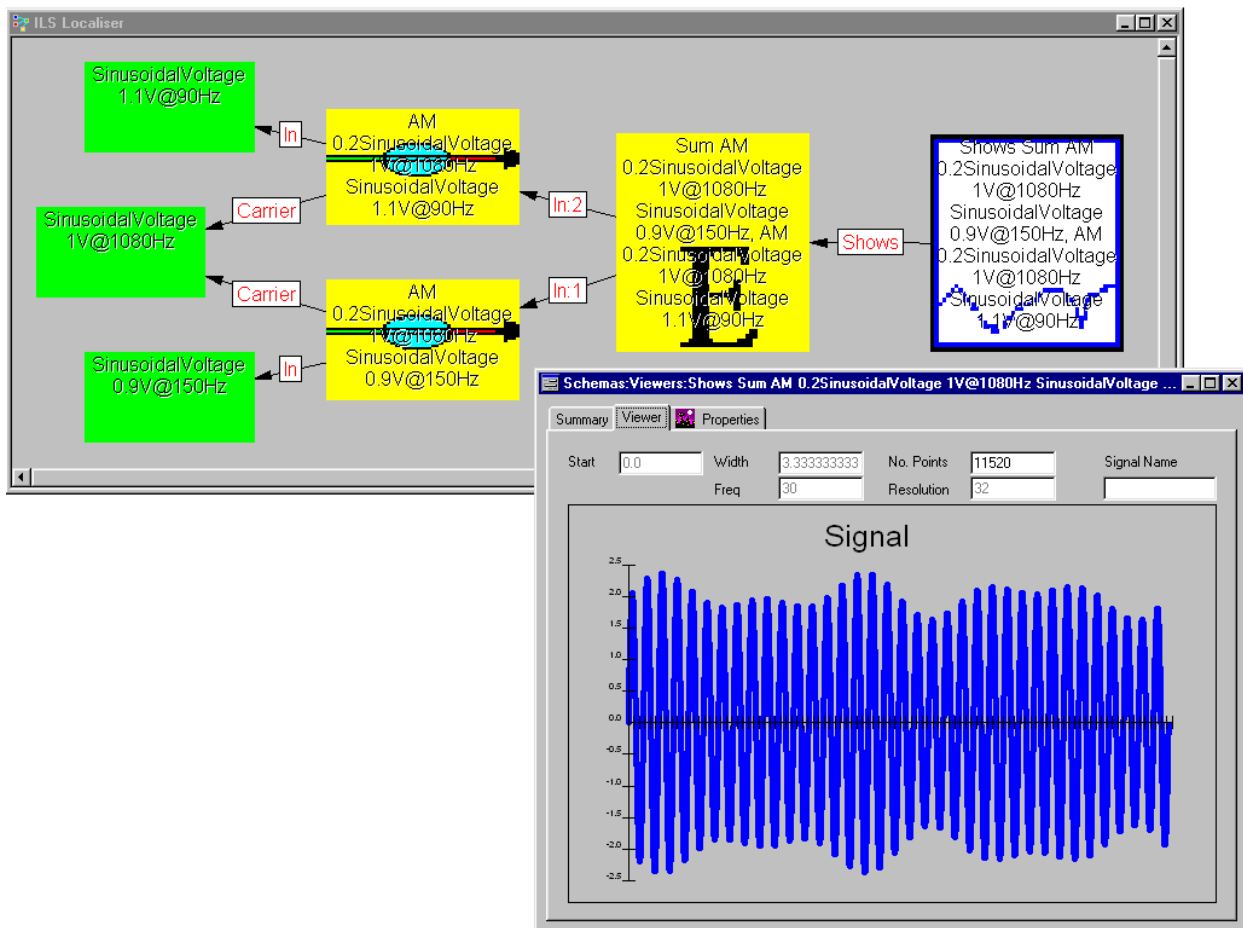


Figure 9. ILS Localiser

Using Signal Graphs it is possible to simulate model equivalents to compare their behavior. Figure 11 shows the two different models of an ILS Localiser (of Figures 9 & 10), both input into a difference component to compare the two outputs. As can be seen from the output graph, there is no difference between these two signals.

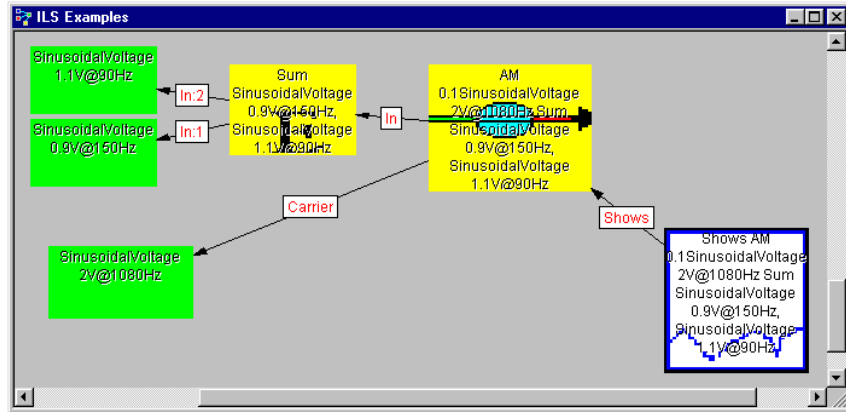


Figure 10. ILS Localiser Model Equivalent

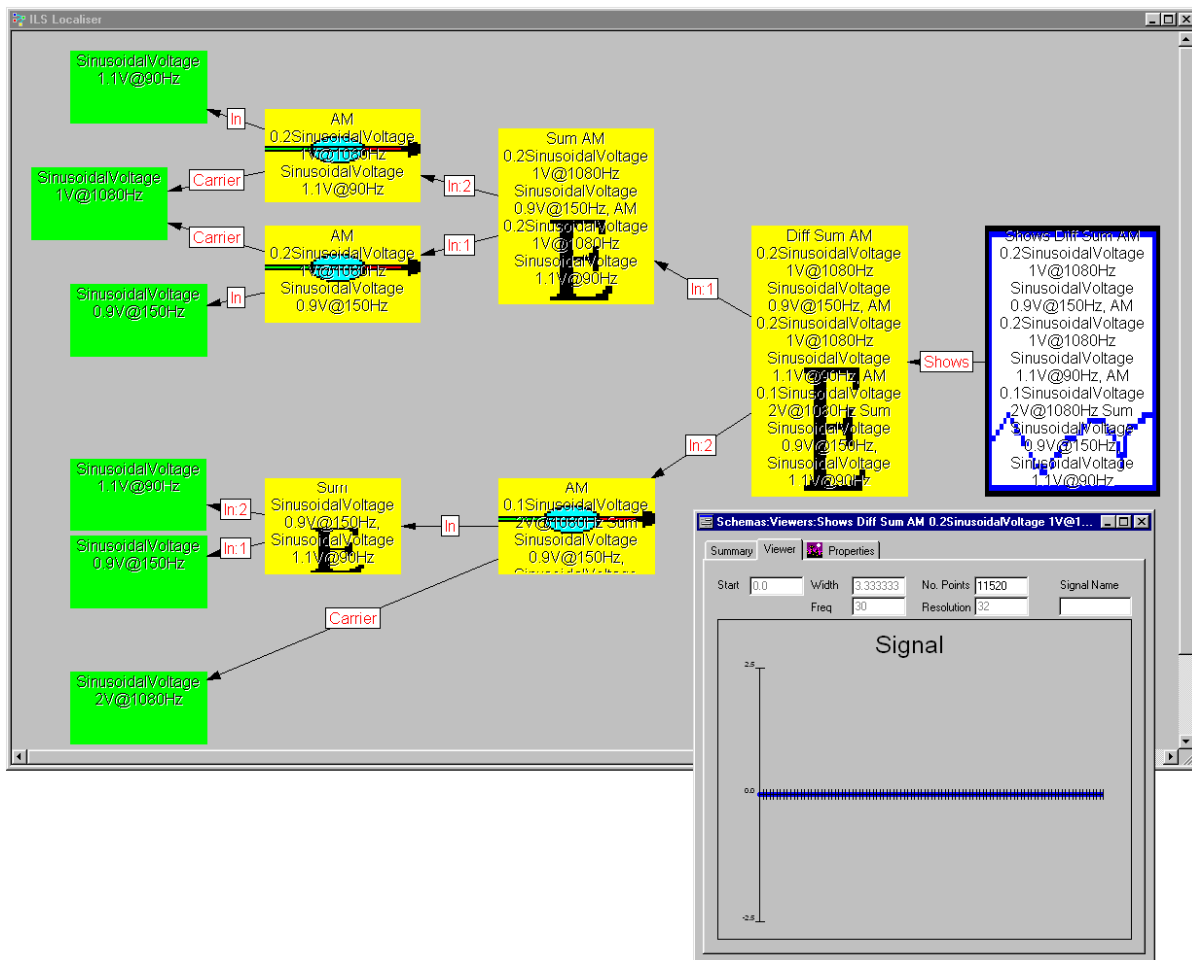


Figure 11. Difference of two ILS Localiser implementations

Having proved the equivalence of these two models, we now find that it is possible to implement the ILS Localiser signal using a resource that would not have been able to implement the original model (Figure 12).

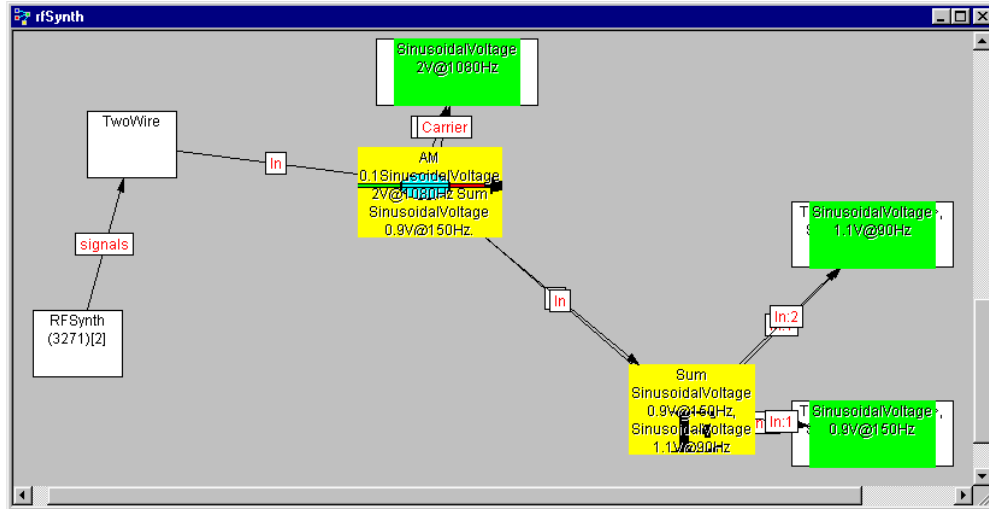


Figure 12. ILS Localiser Implemented on a Racal 3271 Signal Generator

3 SIMULATION

Signal Graphs draw upon a mathematical representation that allows analysis and simulation. A mathematical model is defined using the Signal Modeling Language contained in the SDTD standard. This allows examination of several signal characteristics in time or frequency domain representation, for instance, simulated measurement.

If we look at the mathematical model for a triangular wave, we see that it comprises of three window elements (Figure13). Each window element is a slope of the form, ' $y = mx+b$ ', where the values of the constants are chosen with the width of the window to provide us with our triangle.

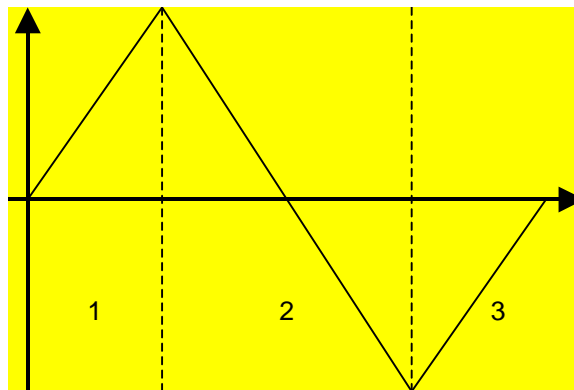


Figure 13. Three Window Elements of a Triangular Waveform

This actual definition, as set out by the standard, is described in Figure 14. Because we have this mathematical model, we can ask questions of our model, such as “*What is your value at time 0.4Sec?*”, or even, “*What are your values corresponding to this series of time values?*”

```

>(\amplitude period -> --Triangle {period=period, level=amplitude}
> let per = fromPhysical period
>     v = fromPhysical amplitude
>     sl = 4.0 * (v / per)
>     qper = per / 4.0
>     wins = Window LocalZero (TimeEvent qper) (linear sl (toPhysical 0.0)) |>
>           Window LocalZero (TimeEvent (2.0 * qper)) (linear (- sl) amplitude) |>
>           Window LocalZero (TimeEvent qper) (linear sl (toPhysical (- v))) |>
>           nullWindow
> in pieceRep (cycleWindows wins)
>

```

Figure 14. SML Definition of a Triangular Waveform

It is the answers to these questions that allow us to simulate all the signals and their interactions. The simulation becomes a quantization of the true signal, with values selected at different points along the independent axis (time).

In addition, ambiguity may be modeled in a practicable manner, where the simulation will show only one of the possible allowed solutions. We can use the simulation to actually investigate effects such as quantization, by plotting a waveform at different resolutions (Figure 15). By differencing our quantized signal with the true signal, we can zoom in to see the magnitude of the quantization errors and identify the resolution of our approximation.

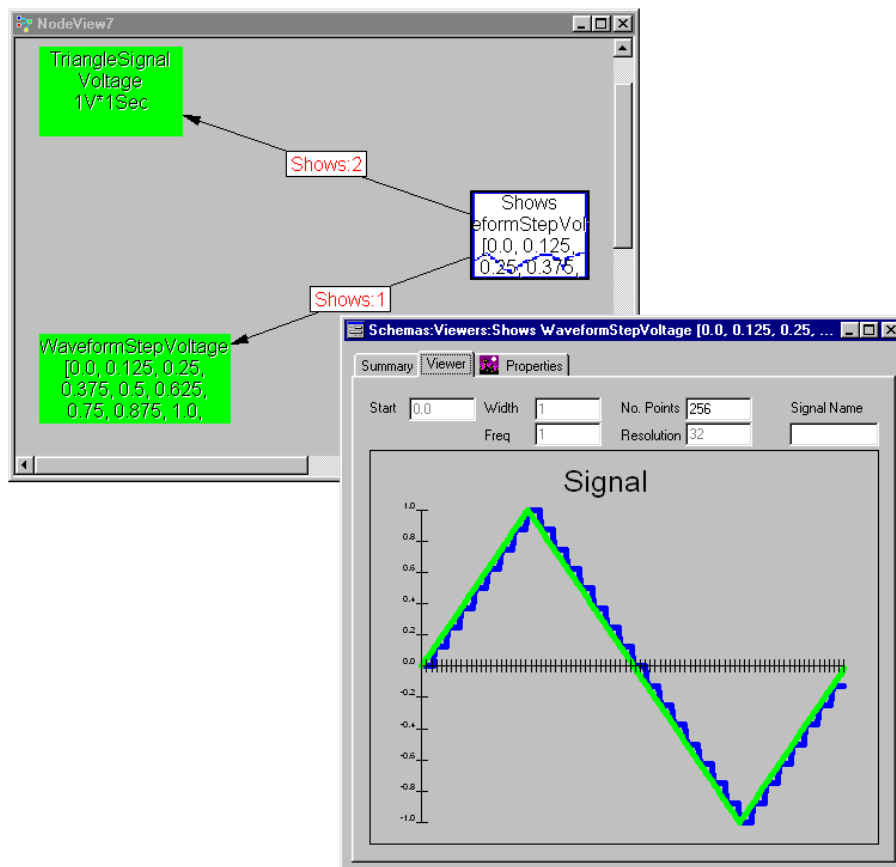


Figure 15. 32 Point Sampling vs 256 Point Sampling of a Triangular Waveform, Showing Quantization Errors

For all bounded, continuous signals we can always reduce our quantization errors by taking a greater number of samples.

We need to be careful to realize that it is the mathematical models that define the signals, not the simulations. However, the simulations do provide a convenient and powerful tool to generate real world signals or compare the real world signal implementations.

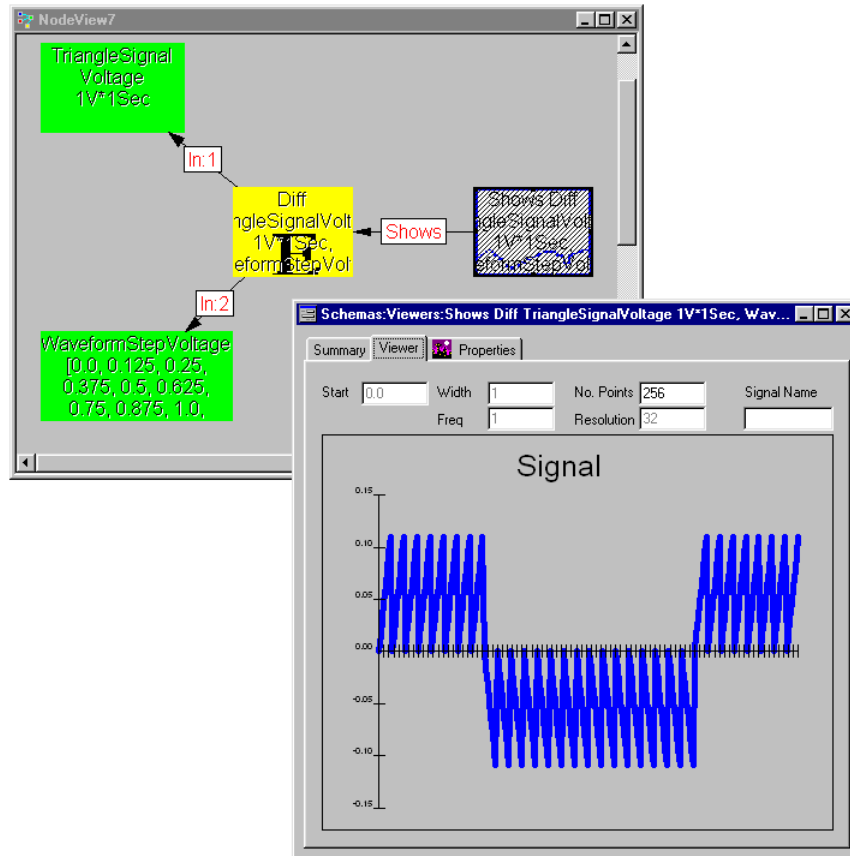


Figure 16. 32 Point Sampling vs 256 Point Sampling of a Triangular Waveform, Zooming in on the Magnitude of the Quantization Errors

4 TEST EQUIPMENT

In order to allow Signal Graphs to be mapped onto general-purpose test equipment, the test equipment exposes its capabilities in terms of which signal components it can support and which configurations are available.

Traditionally, signal based systems have mapped each complete signal description. This is a valid method, but requires exhaustively listing every attribute and condition that the signal could possibly have. When you have a large set of signals, or worse, an extensible signal system, this approach needs to be revised.

We can describe our test equipment in terms of the basic signal components of the Signal Graph. For example, rather than describing an RF Synthesizer in terms of all the signals it can support, we can build up its capabilities in terms of the basic operations it supports and infer complex signal capability (Figure 17).

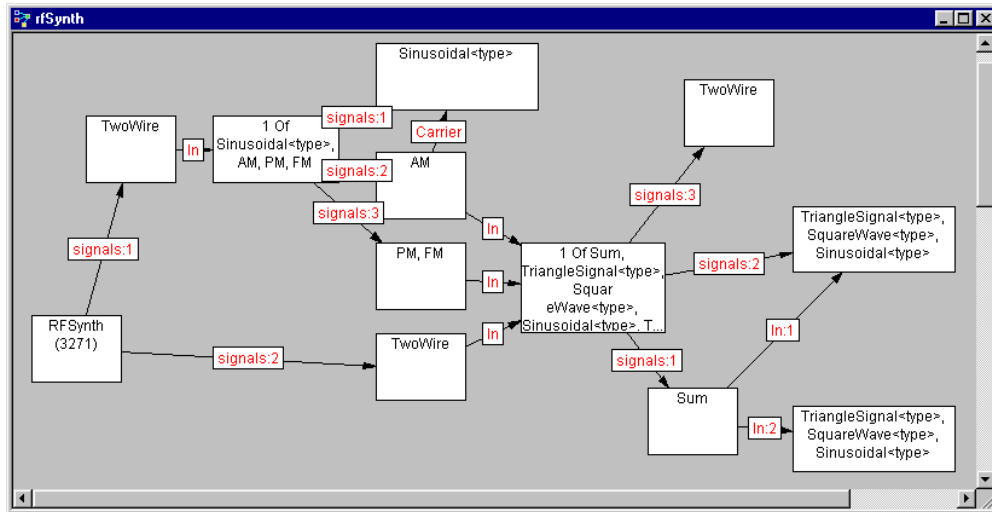


Figure 17. Racal 3271 Signal Generator, Showing ‘One of’

If only simple signals are used, there could be a performance issue in describing known (‘usual suspect’) complex signals. Matching higher order capabilities would be quicker and, at control time, may be able to provide an optimized solution. However, an implementation could determine these higher order complex signals by experience (i.e. “*I have matched one of these before, so the likely-hood is that I can do it again*”), saving the result for future use.

In order to provide alternatives, we need to introduce a new concept of choice, as in choose ‘one of’ the following. For example, in the Racal 3271 Model, we can’t simultaneously use AM and FM. Another feature that this technique provides is when there are limited or shared components, the model of the Signal Generator (Figure 17) identifies that the low frequency output of this device (the internal modulated signal) is seen as a coupled capability. I.e. Either we use the non-modulated signal & low frequency output, or get the modulating signal.

We can use XML to describe this model (Figure 18), providing us with a convenient method of obtaining resource capabilities, without any specialist knowledge of complex signal applications.

Test Instrumentation generally falls in to two classifications; Bounded and unbounded. A bounded instrument description describes a finite number of capabilities. This can be seen by the fact that its description signal capability graph forms a tree. By contrast, an unbounded instrument description potentially describes an infinite number of capabilities. This can be seen by its recursive use of signal capabilities. In the above example (Figure 17), the description was bounded.

```

<Resource name="RFSynth")
  <Signal>
    <Two Wire>
      <One of>
        <Sin>
          . . .
        <AM> . . .
        <PM> . . .
        <FM> . . .
      </One Of>
    </Two Wire>
  </Signal>
</Resource>

```

Figure 18. XML Description of ‘One of’

If we were to try and describe a memory waveform device such as an arbitrary waveform generator, we can see that we could potentially describe an ‘infinite’ number of waveforms, but are constrained by quantization effects and the maximum number of waveform points.

Test resources should describe their capabilities in terms of signal components, implementing their signal control in terms of objects with these interfaces. The best solution is to provide both test resource query capabilities, where we query the test resource for what signal components it supports as well as containing a complete static description of such components.

5 SYNTHESIS

The final realization maps the required signal components against the available exposed capabilities of an ATE. Once the actual resource has been identified, the model parameters are programmed into the appropriate ATE resource and the signal is realized.

Given that we have an individual resource that can supply our signal, we could programmatically ask the resource for each interface, program in the properties and, when we had finished describing our signal model, use a control method (**Run**) to turn the signal on.

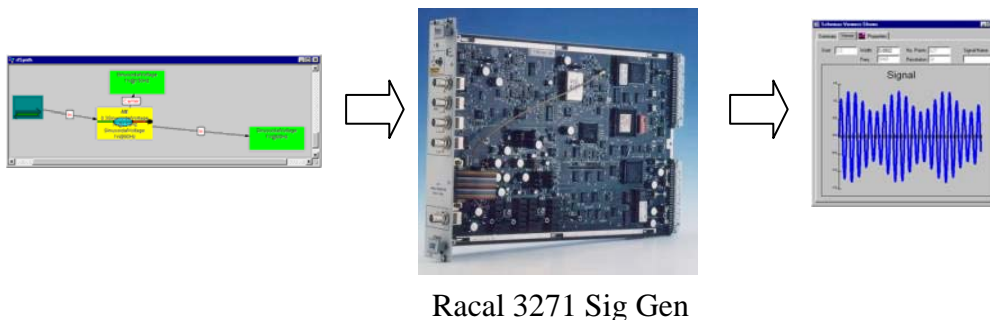


Figure 19. A Signal Graph Mapped to a Single Instrument to produce an AM Signal.

Figure 19 shows an individual resource (in this case a Racal 3271 Signal Generator), programmed from a Signal Graph, to provide an AM signal. Alternatively, our resource may be presented with the ‘complete’ XML Signal Graph model from which to prepare the signal, ready to be turned on.

Where our signal needs to be split across multiple resources, either because the resource does not have the required components or does not have the required resolution or range, we need to assign sub-Signal Graph components to different resource description, plus any necessary ‘glue’ control (Figure 20).

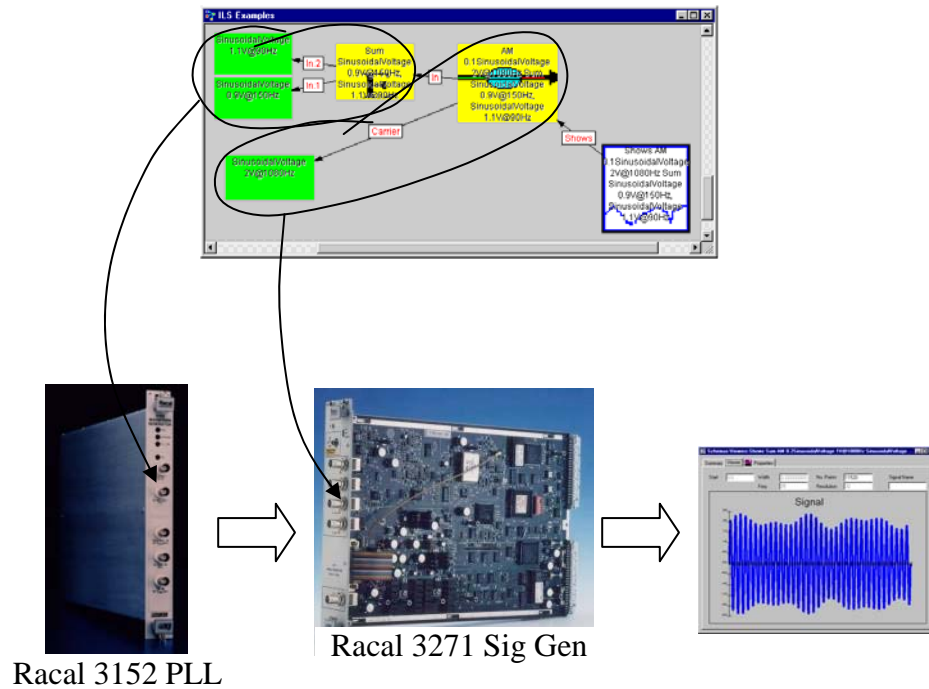


Figure 20. A Signal Graph Mapped to a Multiple Instruments to produce an ILS Signal.

Through pattern matching we can determine a best-fit scenario from a set of resources. Then, using the same interfaces, code our signal either directly (using some chosen programming language) or indirectly (using XML across multiple resources), giving us true Virtual Super Instrument abstractions, using core instrumentation.

6 CONCLUSIONS

Signal definitions created using the IEEE SDTD standard can be both simulated and realized in a real world, physical implementation. The methodology of using the same standard signal model components to describe both UUT signal requirements and instrument capabilities allows user defined signal definitions to be implemented on general purpose ATEs, demonstrating the principle that signals can be reused across different ATE platforms.

Using resource Signal Graph capabilities we can define unique instrument capability in a standard fashion.

7 REFERENCES

- [1] *** IEEE SDTD *specifications*, Draft H, June 2002.
- [2] *** IEEE Std 716-1995, C/ATLAS Test Language
- [3] *** Extensible Markup Language (XML) 1.0 (Second Edition)
- [4] *** VBScript Language Reference, Microsoft, 1997