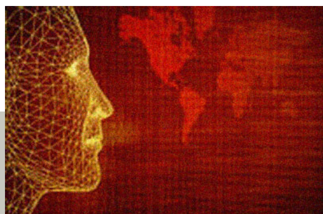


GigaDSP™

D.McClain
SpectroDynamics, LLC

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

LispWorks File View Help Window

Network: Analytic 30 GHz BPSK

Analytic 30 GHz BPSK

1/T = 1 MHz

Location

Paris

Temp [degC]	Pressure [mbar]	Relative Humidity [%]
35.4	961.6	75.3

OK Cancel

GigaDSP

System Sample Rate 6 STOP Panic

Block Browser

- All
- Built-In
  - Annotations
    - Annotation
  - Binary-Operators
  - Bit-Ops
    - Bit-And
    - Bit-Not
    - Bit-Or
    - Bit-Xor
  - Boolean
    - And
    - Nand
    - Nor
    - Nxor
    - Or
    - Xor
  - Clocked
    - Clock-Sequencer
    - Down-Sampler
    - Event-Counter
    - Gaussian-White-Noise
    - Latch
    - Oscillator

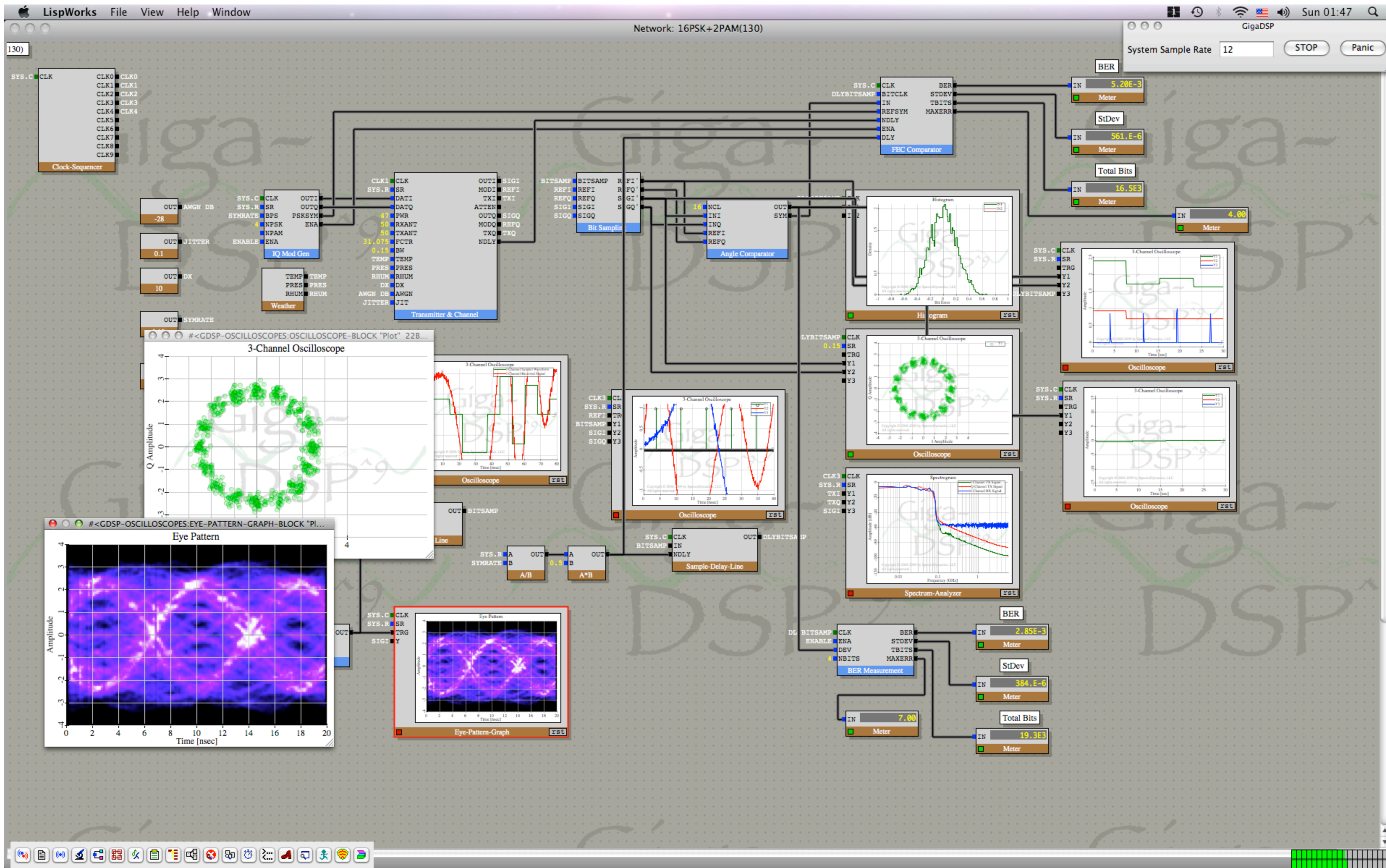
Property-List Editor

Temperature [degC]	37.5
Pressure [mbar]	995.8
Relative Humidity [%]	38.8

Information about the properties

Clock-Sequencer · Oscillator · Weather · Uniform-White-Noise · Sgn · Sub-Clock · Sample-Delay-Line · Impulse Generator · Oscilloscope · Eye-Pattern-Graph · Down-Sampler

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary



Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Block Diagram Modeling

- GDSP allows us to model complex systems
  - Linear and Non-linear systems
  - Transient and Steady-State observations
- Block diagram programming with drag & drop and wire autorouting
- LiveWire^{TM} allows us to modify and rewire network diagrams while simulations are running
- Unlimited macro-block nesting
- New functions defined through combination of existing blocks in macro-blocks, or by writing new Lisp primitives
- Completely asynchronous modeling
- Publication-quality graphics -- cut & paste to documents

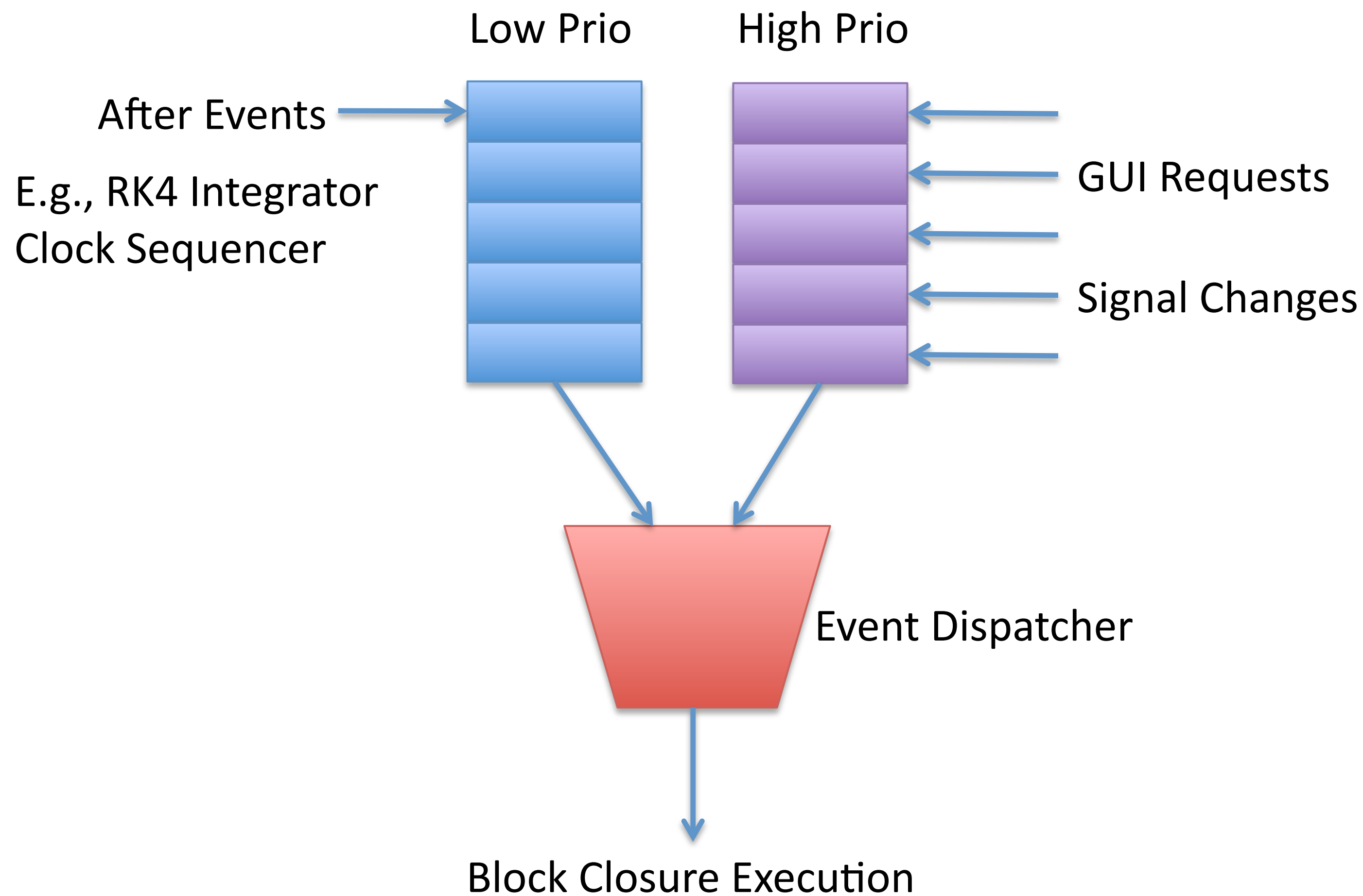
Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Multithreading

- Single simulation action thread
- Multiple interaction threads
  - GUI Interactions
  - Signaling Input Pins (but only when state changes)
- 2-level priority queue
  - Most actions enqueue in the high priority queue
  - Some blocks need After-Actions, which enqueue in the low priority queue
- Clock fires only after all network interactions have ceased
- Blocks are not synchronously clocked
  - But some blocks need a clock edge for sampling inputs

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

2-Level Priority Queue



Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Simulation Executive

- Simulation Executive runs as a separate thread from GUI and others
  - Allows dynamic interaction with running simulations
- Clock fires
  - Causes some blocks to be enqueued in the Hi Prio queue
  - Execution of those blocks propagates signals to other blocks
  - Other blocks that notice a change on their input pins may enqueue themselves in the Hi Prio Queue
  - After all of the signal cascade events, and GUI initiated update events, have been drained from the Hi Prio Queue, then the Low Prio Queue is addressed
  - After each Low Prio execution, the Hi Prio cycle begins anew
  - Finally after all Low Prio events have drained, we begin another clock cycle
- Why do this?
  - How else to get subclock phases within one clock cycle, for clock sequencing and directing deterministic firing in portions of a network?
  - How else to handle Runge-Kutta integration which typically uses ¼ time-step probes and computes their weighted average in each clock cycle?

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Comparison to SigLab

- SigLab was our earlier modeling platform
  - No fancy GUI, mostly text-based
  - Same quality graphics, but no LiveWire™ capability
  - SigLab was a synchronous model
  - Crude macro facility
  - Fewer primitive blocks
  - Feedback networks were clumsy
  - Only one output pin per block

- GigaDSP™ is completely asynchronous
  - Feedback networks very simple
  - Very easy to do multi-rate modeling
  - LiveWire™ changes during simulations
  - Multiple input and output pins
  - Vastly improved measuring devices
    - Oscilloscopes (3-channel, X-Y, various trigger modes, trigger delays)
    - Spectrum analyzers
    - Histograms
    - Transfer Functions (Amplitude, Phase, Group Delay)
    - Meters
  - More primitive functions
    - Filters (IIR & FIR)
    - Analytic Filtering
    - Additional noise sources
    - Waveshapers (User defined, Tchebyshev)
  - More oscillators
    - Sinusoidal, quadrature output
    - Square-wave
    - Triangular wave
    - Ramp
    - Tchebyshev

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Asynchronous vs Synchronous

- Synchronous design is simple to use
  - Every block fires in every time step
  - Firing order is deterministic and easily predicted
  - Many blocks fire when they really don't have anything new to offer
  - Multirate modeling is more difficult
  - Number of computations per clock cycle is fixed by the network under simulation
- Asynchronous design
  - Blocks fire only when conditions change
  - Not every block fires in every time step
  - No determinism in firing order without explicit staging
  - Very easy to incorporate multirate models
  - Two-level priority event queue makes it easy to construct complex processing, e.g., RK4 Integrators, Subclocking, Clock Distribution
  - Clocks are used only for sampling inputs at known instances
  - Number of computations per clock cycle varies, depending on conditions

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Wires and Buses

- Any kind of data can flow along wires
  - Numbers (real & complex)
  - Strings
  - Network packets
  - Images
  - Functional closures
- Mousing a wire shows a sample of the data in a popup text help box
- Buses abbreviate wiring by assigning a name for the signal distribution and eliding explicit wiring
- Buses can clean up messy network wiring
- Moderately sophisticated wire auto-routing to help keep diagrams looking nice (dog-leg routing)

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Input Pins

- Can be set to trigger events when their state changes, or not
- Can be wiring destinations
- Can have constant values assigned
- Special input pins for clock and sample rate
- System Clock auto-assigns to most clock input pins, but manual change is permitted
- Mousing a pin shows a sample of the data on that pin
- Color coding to suggest behavior: clock, rate, signaling, non-signaling inputs

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Output Pins

- Carry the results of block computations
- Any number of output pins per block (not just one)
- Can have initial conditions assigned
- Can be assigned to buses
- Wiring is one (output pin) to many (input pins)
- Mousing a pin shows a sample of data on that pin

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Block Firing

- When one or more input pins detect changed conditions, for signaling input pins
- When a clock pin is triggered (always triggers)
- System clock is a pervasive block hidden from view
  - System clock bus SYS.C
  - System sample rate bus SYS.R
  - Auto-assigned when new blocks are dragged to the design area, but can be changed

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Macro Blocks

- Can be defined at any time
- User specified input / output pins
- Drag any blocks into the macro interior
  - Primitive blocks
  - Other macro blocks
- Serve as sub-networks that appear as a single block to the outer level
- Can be nested ad infinitum
- Standard Library of Macro Blocks, plus user defined
- Can be first line of modeling before committing to Lisp coding for a new primitive block
- Full-fledged networks in their own right, except that clock input pins are not automatically assigned to System Clock
- Macro blocks can also clean up network appearances by placing related sub-groups together into one macro block

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Property Editors

- Highlighted block brought up in property editor to permit internal parameter changes
- Can also right-click a block to bring up a property editor
- Properties:
  - Numeric parameters with constraint checking
  - Choice lists
  - Plotting parameters
  - Fonts and Labels
  - File selection dialogs
  - Etc…

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Block Browser

- Presents primitive blocks in a number of categories
- Blocks may occupy several categories
  – Measurements, clocked, math, etc.
- Top-level categories expand to expose inner sub-categories, and ultimately primitive blocks
- Drag & drop to network design area

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

System Clock

- Pervasive, but hidden block
  - Buses pre-assigned SYS.C and SYS.R
- Carries system clock and sample rate to networks
- Settable by the user with the SampleRate box
- Despite being asynchronous, the clock allows a discrete time simulation with a known sample rate
- Clocks are not observable entities, e.g, in an oscilloscope or spectrum analyzer
  - Why? …

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Example of Existing Primitive Blocks

- Measurement
  - Oscilloscope, Spectrum Analyzer, System Transfer functions (amplitude, phase, frequency, group delay), Histogram, Eye-Pattern, Meters
- Noise sources
  - Uniform, Gaussian, Brownian, 1/F
- Math – everything you could imagine, incl. FFT's, RK4 Integrators, Convolutions, Correlators
- Logic and bit-level blocks
- Filters
  - IIR, FIR, Analytic, User defined FIR, User defined IIR,
  - Butterworth and Bessel 2nd Order HPF, LPF, BPF, BRF, APF,
  - 1st order LPF, HPF, APF
- Subclocks, delay lines, 1/Z unit delay, clock distribution, latches, decimators, gates
- Comparators rising edge, falling edge, threshold crossing
- Limiters & clippers
- Network input and output ports
- Gray coding, FEC (Reed-Solomon, and others)
- Event detectors and counters
- Units conversion (Amplitude, angle, frequency, time, length)
  - E.g., dB, turns, radians, MHz, GHz, ns, ps, km, miles, etc
- Debug output panes

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

The Power of Lisp

- All of GigaDSP™ is written in Lispworks Lisp
  - Except for FFT's – uses state of the art external lib
- Runs on Windows XP, Vista, Mac OS X
- High performance, natively compiled Lisp
- Hyper-rapid prototyping with block diagrams
  - Easy to translate networks into new primitive blocks when deemed necessary
  - Special DEFBLOCK macros makes it all so easy
- CLOS used liberally
  - Importance of method combination
    - NCONC, PROGN, Standard (:before, :after, :around, and direct)

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Statement of the Problem

- We need to assure QOS in RF networks
  - Depends on weather conditions
  - Depends on distance from xmtr to rcvr
  - Depends on xmtr power levels
  - Depends on antennae
  - Depends on noise sources
  - Depends on obstructions between xtmr and rcvr
  - Depends on data rate
  - Depends on modulation techniques
  - Depends on use of FEC

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Solution through Simulations

- We can build up progressively more complex, and hence, more complete, models of systems
- We can try various methods for improving our RF data technology
- Using GDSP this can be done very rapidly, without writing a single line of code, or committing to hardware for evaluation
- We can vary conditions in the model to probe the boundaries of proposed solutions
- Much cheaper to model, in both time and cost, than to build and test
- Build and test only after we have narrowed the field through simulations

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Complexity

- As models become more complex they take longer to produce an answer
- Answers only become apparent after many many trial runs with varied conditions
- BER = Bit Error Rate
  - Varies with conditions, generally known as SNR
  - Can be computed for several different values of SNR and then extrapolated
  - Problem is… *it takes 2 hours with GDSP to produce one estimate for given conditions at one SNR level for* $BER<10^{-5}$
  - *… And we need* $BER<10^{-9}$ *or better!*

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Do we go to "C"?

- Dare we go to C for system speedup?
  - How much faster than good compiled Lisp?
    - Maybe 2x at most, probably only 20-30% better
    - Still not enough improvement to make much difference
- How long would it take *(assuming we were stubbornly bone-headed)* to write the system in C?
  - Line count of active lines of code *(no blank lines, no comment lines)* shows 100 KLOC of Lisp.
    - That translates into at least 1 MLOC of C *!!*
  - Maybe 10-100 MY (1 MY = 1 man for 1 year)
    - depending on *"cowboy"* status and experience
  - What about all the bugs introduced?
    - Even I can't write bug free C code, as maddening as that is, after having authored C compilers and having personally written more than 5 MLOC in C and C++.
- Measurements and experience shows that:
  - In C, about 1 bug every 10 lines of code
    - I'm not talking about stupid beginner's bugs like wild pointers and such..
    - I'm talking about all the distractions imposed on the programmer by the language so that he/she loses sight of the overall algorithm
  - In Lisp, about 1 bug every 50-100 lines of code
    - Algorithms don't work as well as anticipated, maybe even goof up their implementation
  - Lisp is roughly 10x more expressive than C *(…at least!)*
  - Hence, bug count in C is likely to be 50-100 times worse than what we already experienced in building GDSP – *YIKES!*
  - IMPOSSSIBLE in terms of cost and time
- Also… Remember Greenspun's 10th
- Bottom Line – *NO!*

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Throw More Computers at the Problem

- Distributed computing will help by factors of 10's to 100's, depending on size of farm
  - Traditionally, Distributed Computing is a very tedious and error prone process
- Butterfly™ makes this as easy as writing MT programs on one computer
  - The same code works, whether distributed or not
  - The code is oblivious to the location of cooperating processes
  - Message passing paradigm, not shared global memory
- Okeanos™ database makes it possible to harvest data, retain for further study, and share between widely separated test sites
  - Not hampered by need to force our data into tables for SQL
  - Data types can evolve with changing models without losing what we already collected
  - We don't want to become database maintainers, constantly updating and sharing schema changes, and then rebuilding data tables
  - OKNO lets us evolve smoothly and painlessly

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Parallel Processing

- I need 5 data points, each costing >2 hours of runtime
  - Use 5 computers, each running different ambient conditions
  - Trouble is… it still takes >2 hours to get one, or five, data points
  - What if something wrong in model? Won't find out for 2 hours??

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Pipeline Processing

- Suppose model processing chain consists of 5 chunks:
  - Data and noise generation
  - Encoding and modulation
  - Channel estimation
  - Demodulation and decoding
  - BER estimation
- Split these five sections into separate computers for 5x speedup
- Now, for one set of conditions, it only takes a half hour to find out if we made a mistake somewhere

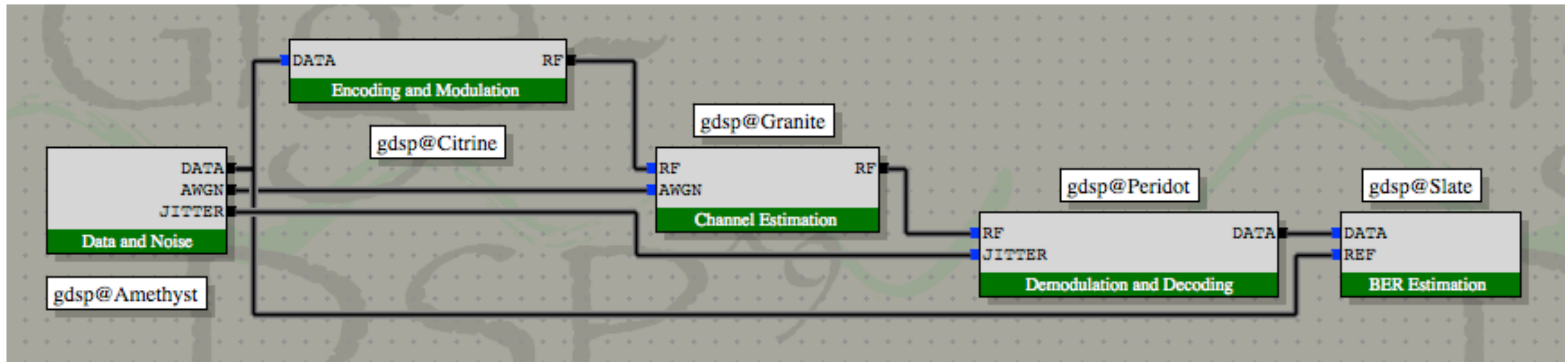
Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Parallel or Pipeline?

- Parallel processing doesn't really need the sophistication of Butterfly™, just Okeanos™ so that we can harvest the data.
  - But OKNO needs BFLY anyway, so that all 5 computers can aggregate their data…
- Pipeline processing more sophisticated and needs the special capabilities of Butterfly™ directly
- Both methods take the same amount of time to generate 5 data points
  - But we work more effectively when we pipeline

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary

Pipeline for BER Estimation



Each of the wires represents a Butterfly™ connection between machines, one processing block per machine.

Note that this is not just a simple pipeline. Many blocks require data generated by two prior blocks. The final BER Estimator requires the same data stream as used through the model for reference.

All wires carry actively changing information, not static setup parameters.

Copyright (C) 2009 by SpectroDynamics, LLC
Company Proprietary