

01000111 01100101 01100100 01100001 01100101
01000111 01100101 01100100 01100001 01100101
01000111 01100101 01100100 01100001 01100101
01000111 01100101 01100100 01100001 01100101

Gedae 4.5 Release Notes

May 2004

Address: Gedae, Inc.
18000 Horizon Way, Suite 200
Mt Laurel, NJ 08054
Telephone: (856) 231-4458
FAX: (856) 231-1403
Internet: www.gedae.com

1 New Features

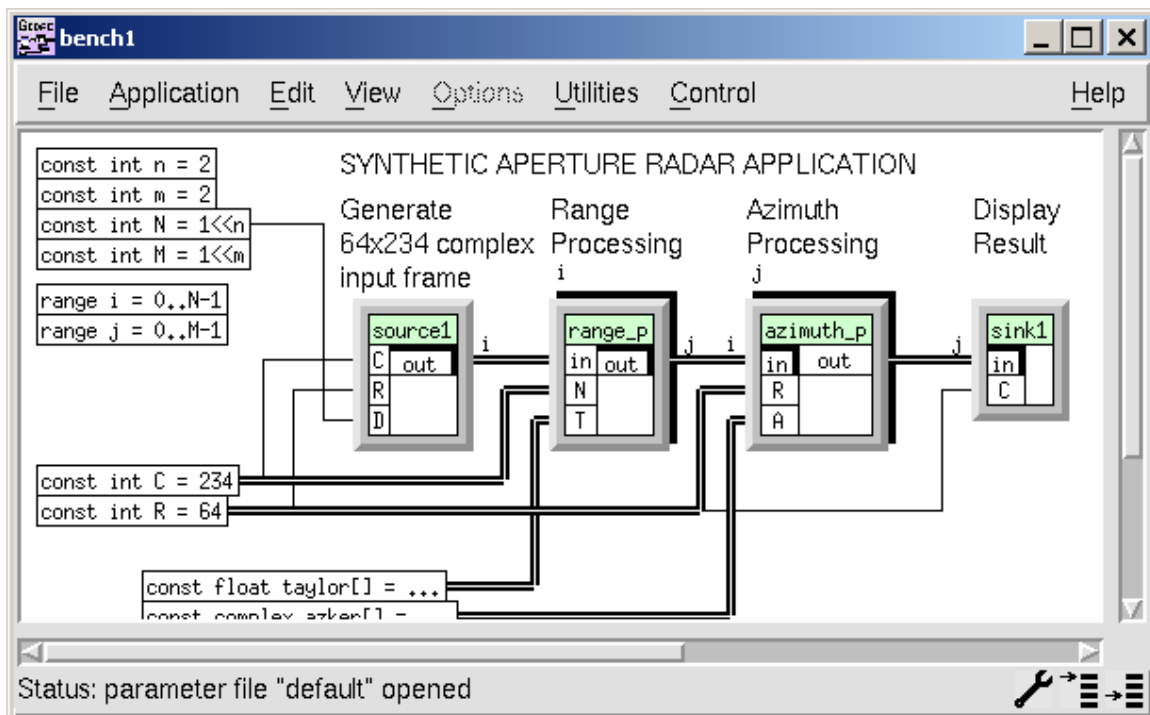
Geda 4.5 has a number of new features. These include:

- A feature to automate the process of subscheduling graphs.
- An extension of the method provided in Geda 4.3 for improving compilation times of target processor executables.
- 40 new primitives added to the Geda library.

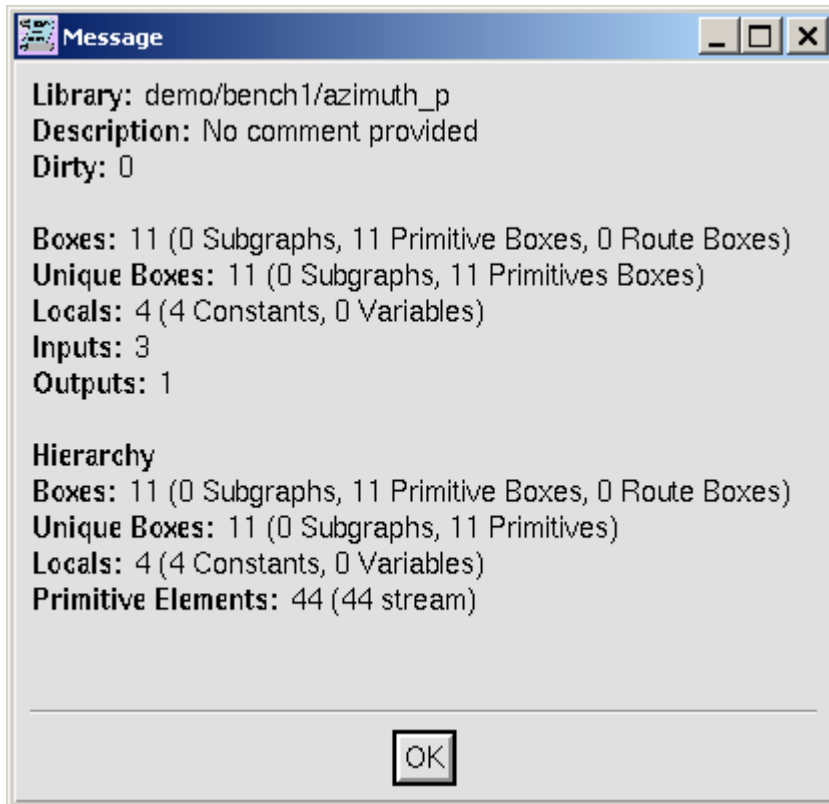
The complete list of the new features follows. In prior releases the features and bug reports were labeled “SCR”, note that the label has been changed to “Case” and new numbers have been assigned to each one.

Case 1049: Info Dialog Enhancement

The information given when a hierarchical box is clicked on in the Box Info mode has been enhanced. The information now includes the total number of box elements instantiated within that hierarchical group. This count includes the multipliers resulting from using families and provides the best estimate of the complexity of the instantiated graph.



For example, clicking with the information cursor on the `azimuth_p` box pops up the message dialog:



The new field "Primitive Elements" indicates that there are 44 primitive box elements in the `azimuth_p` subgraph. The 11 boxes inside the `azimuth_p` subgraph times the size of the family index j equals 44 primitive box elements.

Case 1064: Automated Subscheduling

A method for automatically creating subschedules has been added to Gedae. The tool creates a hierarchy of boxes based on the box gains and shows what set of boxes can be effectively subscheduled. The tool automatically creates subschedules based on this hierarchy to keep schedule sizes under a user selectable minimum. The tool also allows the user to individually set the level of subscheduling on individual members of the hierarchical primitive grouping. A detailed description of subscheduling and the automated subscheduling tool is available in the new manual on subscheduling. The manual is located at <http://gedae.com/SUPPORT/Manuals/SubschedulingDescribed.pdf>.

Case 1067: NT Audio Primitives

Primitives are added for reading and writing WAV audio files. The primitives use the libsndfile library. The NT version of this library is shipped with Release 4.5; Linux and Solaris users must download and install the library themselves if they wish to use the new primitives. The following new primitives are in the `embeddable/stream/audio` directory.

<code>c_wavwrite</code>	Write WAV file using char data
<code>s_wavwrite</code>	Write WAV file using short data
<code>wavwrite</code>	Write WAV file using float data
<code>s_wavread</code>	Read (repeatedly) WAV file as shorts
<code>s_wavread1</code>	Read (once) WAV file as shorts
<code>Wavread</code>	Read (repeatedly) WAV file as floats
<code>wavread1</code>	Read (once) WAV file as floats

Case 1072: Primitives Useful in Vector/Matrix Feedback Loops

Currently to implement a feedback loop on a stream of arrays, one must use `m_s` and `s_m` (or `v_s` and `s_v`) boxes to set the size (or non-inplace boxes like `m_resize`). This operation can be completed in one box, so the following new primitives have been added to the library.

	<code>v</code>	<code>vi</code>	<code>vx</code>	<code>vz</code>	<code>m</code>	<code>mi</code>	<code>mx</code>	<code>mz</code>
Init size - inplace box that has input parameters for the size	<code>v_initsize</code>	<code>vi_initsize</code>	<code>vx_initsize</code>	<code>vz_initsize</code>	<code>m_initsize</code>	<code>mi_initsize</code>	<code>mx_initsize</code>	<code>mz_initsize</code>
Init delay - sets a delay of 1 where a parameter sets the first token	<code>v_init_delay1</code>	<code>vi_init_delay1</code>	<code>vx_init_delay1</code>	<code>vz_init_delay1</code>	<code>m_init_delay1</code>	<code>mi_init_delay1</code>	<code>mx_init_delay1</code>	<code>mz_init_delay1</code>

Case 1073: `init_delay` Primitives for All Data Types

Versions of the primitive `embeddable/stream/init_delay` have been added for the integer (`embeddable/stream/integer/i_init_delay`), complex (`embeddable/stream/complex/i_init_delay`), and `splitx` (`embeddable/stream/splitx/z_init_delay`) datatypes.

Case 1079: View Only License

A new license type has been added to allow users to display Gedae graphs and settings on the terminal without editing, saving, or running. This license is intended for use in peer level reviews. It allows the sharing of information contained within the Gedae graph without requiring a license needed to develop, run, and deploy the applications.

Case 1091: Target Processor Compilation Time Improvement

The same changes made to Gedae 4.3 to enhance the compile time of host primitives has been made to enhance the compile times of target primitives. For host primitives prior to Gedae 4.3, `gmake` was used to check the dates of primitives against the generated `.c` and compiled object files. As of Gedae 4.3, Gedae directly checks the file times including a search of the include paths and determines which primitives must be code generated to `.c` files. It also determines which `.c` files need to be compiled to object files. Based on these determinations the following two script files are created: `gen_files` and `compile` (or `genfiles.bat` and `compile.bat` on NT). The `gen_files` script code generates the `.c` files and the `compile` script compiles the `.c` files into `.o` files. Gedae 4.5 now avoids calling `gmake` to make the target primitive `.o` files but instead creates a script that compiles all the primitives for the target.

Case 1103: `E_` Functions and Primitives for Short/Char Conversions

`E_` functions and primitives have been added for converting from signed/unsigned shorts/chars to floats. The new `E_` functions are:

<code>e_avfloat(char *a,int ia,float *c,int ic,int n);</code>	Convert from chars to floats
<code>e_uavfloat(unsigned char *a,int ia,float *c,int ic,int n);</code>	Convert from unsigned chars to floats
<code>e_svfloat(short *a,int ia,float *c,int ic,int n);</code>	Convert from shorts to floats
<code>e_usvfloat(unsigned short *a,int ia,float *c,int ic,int n);</code>	Convert from unsigned shorts to floats
<code>e_uivfloat(unsigned int *a,int ia,float *c,int ic,int n);</code>	Convert from unsigned ints to floats
<code>e_vafix(float *a,int ia,char *a,int ic,int n,int f);</code>	Convert from floats to chars
<code>e_vuafix(float *a,int ia,unsigned char *a,int ic,int n,int f);</code>	Convert from floats to unsigned chars
<code>e_vsfix(float *a,int ia,short *a,int ic,int n,int f);</code>	Convert from floats to shorts
<code>e_vusfix(float *a,int ia,unsigned short *a,int ic,int n,int f);</code>	Convert from floats to unsigned shorts
<code>e_vuifix(float *a,int ia,unsigned int *a,int ic,int n,int f);</code>	Convert from floats to unsigned ints

The new primitives (underneath the embeddable directory) are:

	Convert to float	Convert from float
sc	stream/char/sc_s	stream/s_sc
suc	stream/uchar/suc_s	stream/s_suc
ss	stream/short/ss_s	stream/s_ss
sus	stream/ushort/sus_s	stream/s_sus
sui	stream/uinteger/sui_s	stream/s_sui
vc	vector/char/vc_v	vector/v_vc
vuc	vector/uchar/vuc_v	vector/v_vuc
vs	vector/short/vs_v	vector/v_vs
vus	vector/ushort/vus_v	vector/v_vus
vui	vector/uinteger/vui_v	vector/v_vui
mc	matrix/char/mc_m	matrix/m_mc
muc	matrix/uchar/muc_m	matrix/m_muc
ms	matrix/short/ms_m	matrix/m_ms
mus	matrix/ushort/mus_m	matrix/m_mus
mui	matrix/uinteger/mui_m	matrix/m_mui

2 Bugs Fixed

The following is a list of the bugs that have been fixed.

Case 1036: Distributed Segmentation Double Reset

Double reset can occur on a segmented schedule that is on a different processor than the segmenter. The problem can be observed on the Trace Table in a schedule SrcFire queue. The SrcFire queue shows two reset markers on the queue at the beginning of a segment. This problem has been fixed.

Case 1050: Setting Graph Parameter Array Values on NT Does Not Work

Setting graph parameter array values on NT using "filename xyz" format doesn't work. The problem is that the binary files were opened using `fopen("xyz", "r")` but on the NT binary files must be opened using `fopen("xyz", "rb")`. Otherwise when an EOF character is found, the file read terminates. The problem only occurs on NT and the symptoms are that an array that is specified to be a particular size is only partially read - the remainder of the array is uninitialized

Case 1051: Graph Will Not Schedule if Branch Statically Follows Merge

If there is a circular dependency in a graph in which Box A statically follows Box B and Box B dynamically follows Box A, then the graph won't schedule. Normally Box A, which dynamically precedes Box B, must fire before Box B. This rule is needed to make normal branch merge constructs schedule reliably. However, static dependencies need to take precedence over dynamic dependencies. The problem has been fixed in the given situation by forcing Box B to fire first, breaking the deadlock.

Case 1053: Terminate Standalone VxWorks Process and Run Destroy Methods

If a launch package is generated to run without a host, then there is no way to run box terminate methods to free resources on reboot. The standalone processes can now be terminated under primitive control by calling the function:

```
void embExitSA(void);
```

So for example, a primitive `Apply` method can, based on an internally calculated condition, call `embExitSA`, which terminates the process.

```
Apply: {  
    ..  
    if (in[0] > 1000) {  
        embExitSA();  
    }  
}
```

When called from a standalone target executable, this function runs all the primitive terminate and destroy methods on the calling target processor and then runs the BSP `embExit` routine to finish the cleanup. When called from a primitive under the control of a host command program, the `embExitSA` function does not cause the command program to exit but rather calls `embTerminateNormal` ("simulating standalone exit").

Case 1054: Allocation Failure During Schedule Creation Not Obvious

When an allocation error occurs during schedule creation, an error is printed to the screen and sometimes an assert dialog pops up. This has been fixed, and now the Gedae error dialog pops up and displays information about the allocation failure. Also, the same information that was printed to the screen is displayed in the error dialog.

Case 1055: Need to Override Host IP Address Host Sends to VxWorks Target

The host sends its IP address to the VxWorks target. For some network configurations the VxWorks target knows the host by a different IP address. The solution is to override the host IP address sent to the VxWorks target. This override is implemented as an additional argument for the user to set in the embedded config file. In the processor description section, the user is now able to specify the VxWorks processor type as:

```
vxworks: {  
    Type: "evxworks"  
    Make_Params: "evxworks/runtime_make_info"  
    Info: "-ip 192.168.2.32"  
    Memory_Desc: {}  
}
```

where "-IP 192.168.2.32" will set the hostIP address sent to the target to be 192.168.2.32. If an -IP param is not present, then the host will look up its IP address.

Case 1074: Dy4AV2 Installation

The first installation of the Gedae Dy4AV2 BSP was performed at a customer's site. Several bugs in the installation procedure were fixed, and the lessons learned and common errors were recorded in the Gedae Dy4AV2 installation manual.

Case 1090: Bytes Allocated for External State Proportional to Square of Element Size

If an external state vector is declared as:

```
<Type> out[N]
```

where <Type> is any type definition (including standard C types), then the number of bytes allocated for the external state is:

```
N*sizeof(<Type>)*sizeof(<Type>)
```

rather than

```
N*sizeof(<Type>)
```

the latter being correct. This problem has been fixed.

Case 1093: No-host Launch Package Target Execs Can't Run Twice on VxWorks

When created with the no-host option, a launch package can't run twice on VxWorks without being reloaded. This problem has been fixed.

Case 1097: Gen Problem in Variable Matrix Merge Box

A box that uses the max function to calculate the third maximum value of an output stream from and input family of third maximums failed with a segfault. For example:

```
Input: {
nondet stream float
[i:N]in[n[i],Max0[i]][m[i],Max1[i]]<[MaxC[i]]>;
stream int c;
}
```

```
Output: {  
stream float out [p, max (Max0, N) ] [q, max (Max1, N) ] < [max (MaxC, N) ] >;  
}
```

The `max (MaxC, N)` in the output declaration caused the problem. This problem has been fixed.

Case 1102: `set_ptr` Function Doesn't Work Correctly with Multiple Boxes of Same Class

The `set_ptr` function doesn't work correctly if there are two function boxes in a graph of the same class with a pointer output. This problem has been fixed.

Case 1107: Gen Cannot Handle Spaces Before a Parenthesis in Built-in Functions

When using the `consume` function in the following way:

```
consume (a, b);
```

In Gedae 4.4, Gen returns an error and does not read in the primitive. Gen will now allow extra spaces before the parentheses when using built-in functions like `consume`.

Case 1016: If Graph Cannot Be Opened in FG Editor, Open Graph in \$EDITOR

Sometimes there are syntax errors in graph files that prevent the parser from opening them. If this happens, pop-up an `$EDITOR` with the error inside, allow the user to fix the error, and then retry opening the graph.

Case 1112: Disallow Eval from Having a Local Section

Eval boxes should not retain state information. Thus, they should not have `Local` sections. Previously, Gedae would allow eval boxes to use `Local` sections, but would not allocate the space or setup the data pointers, which could lead to segfaults when the data was used. Now, Gedae disallows eval boxes from having a `Local` section.

Case 1117: Non Standard Parameter Types Are Byte Reversed on Target Processors

Non-standard parameter types are byte reversed on target processors that have the opposite byte ordering of the host. For example an eval box that defines an int32 type as

```
typedef int int32;
```

demonstrates the problem when connected to a similar static input of a box mapped to a byte reversed target. The problem has been fixed by putting all unknown types that have sizes that are multiples of 4 bytes (the size of an integer) through the host to target integer conversion routine. This fixes the problem on all current Gedae BSPs but will not fix the problem for BSPs in which the conversion routines for 32 bit words are not constant for all data types. For example, a BSP from a host using IEEE formatted floats communicating with a target using TI floating point format will not be able to convert structures containing floats as elements. It will still be able to convert floating-point streams and parameters correctly.

Case 1124: Disallow Saving Primitives When Name Does Not Match Filename

The primitive editor will no longer allow the user to save a primitive when the Name section of the primitive does not match the filename of the file. When the primitive editor allows such a save, the box cannot be reloaded into Gedae due to the name mismatch. This problem has been fixed.

3 Known Bugs

Case 1024: Accommodate Different Exceed Versions in `initGEDAE`

In order for Gedae to work with Exceed/XDK 9.0, `makeGEDAE` had to be modified. This could be accomplished by a question or parameter to `initGEDAE`.

Currently users must view the FAQ for 7.1 changes
http://www.gedae.com/SUPPORT/FAQ/exceed_7_1.html
and then remove `xlibcon` from `makeGEDAE` and `ent` files mentioned in the FAQ.

Case 1026: Broadcast Transfer Mechanisms

Currently all Gedae data communication is point to point. However, many target processors support efficient broadcast mechanisms. The goal of this task is to extend the Gedae BSP API to allow broadcast data transfers to be part of the BSP. A second goal is to fix the DSA mechanisms that cannot be implemented when there is fanout (see Case 1149). Rather than fixing that problem directly, a broadcast DSA capability will be implemented.

Case 1056: FGU of Hierarchical Typedef Boxes

FGU does not transfer hierarchical typedef boxes correctly. The typedef used to define the input of the box is set to the old directory rather than the new.

Case 1057: Graph Stalls

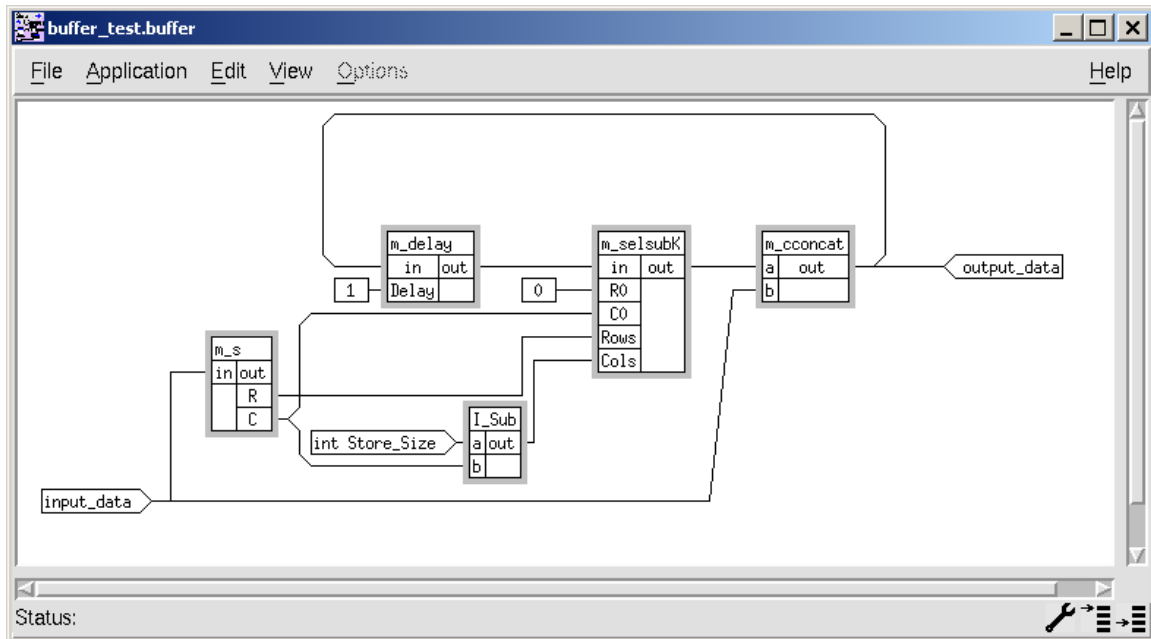
A rare condition can cause a graph to stall (or segfault) in the case of the controlled static schedule being partitioned to two processors in the form

A->B->A

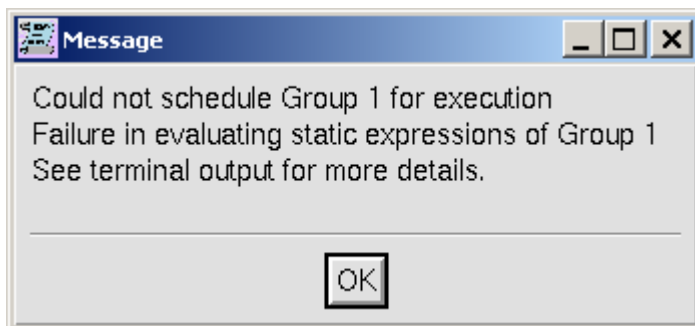
The problem scenario is that the schedule is partitioned into three parts with both the beginning and ending parts to the same processor. In this case, Gedae splits the two parts scheduled on processor A into two different static schedules numbered `n.1` and `n.2` (for example 2.1 and 2.2). The problem is produced when the processing at the tail end is slower than the processing at the front end, which causes the control message queue to back up and overflow. The condition is rare because the problem only happens when the graph is not keeping up with the input data rates.

Case 1071: Dimensions Not Set Correctly

The dimensions are not set correctly in the graph below due to an interaction between the output parameter from the `m_s` box and the propagation of the dimensions around the feedback loop.



The error manifests itself as:



The workaround is to directly set the `m_selsubK` input parameters from graph parameters rather than from the `m_s` output.

Case 1078: Geda Overwrites Protected Files

Geda overwrites files that are protected as a result of being checked out of a CMS. It causes a problem because it defeats the CMS system.

Case 1108: Setting Default Subschedule Queue Capacities Correctly

The queue policy and capacity of a queue feeding a subschedule may not be set correctly. If one or more queues feed the top-level static schedule, then all the queues should be set to have a queue policy of parent instead of nreq_q. If one or more queues feeds a subschedule at level n and no queues feed subschedules at higher levels, then all the queues should be treated as though they feed the schedule at level n. In this case all queues should have a default dest_qp of nreq_q and their default capacities should be such that the level n schedule will be able to fire.

Case 1122: Embedded Build Can Require a `makeGEDAE CLEAN`

If an application is repartitioned, then the target executables don't get relinked. The problem is all the .o files are older than the targets, and the fact that there is a new link line does not force the target library and target executables to rebuild.

Case 1127: Static Schedule Gain Setting Algorithm Propagates Through Dynamic Queues

If there is a dynamic queue internal to a static schedule, then the gain setting algorithm propagates the gain through the dynamic boundary. Since the decimate/interpolate rates set for such boundaries are not necessarily a reflection of the actual data flow gain, these rates can lead to a report of a gain error in the graph

Case 1139: Unterminated Comments

Unterminated comments cause the Gedae parser to segfault.

Case 1140: Parser Problem

The Gedae parser doesn't handle an odd number of quotes (") well.

Case 1141: Problem with Variable Vectors and Delay

If a dynamic variable vector input is preceded by a `vv_delay`, then the graph segfaults during scheduling.

Case 1142: Arrays of Strings Not Allowed

Gedae currently allows string array graph parameters to be declared as:

```
const string X[] = {"hello", "world"}
```

or

```
string X[i] = [i]Y
```

where Y is a family of strings.

In either case, the values so declared are not correctly set, and therefore, should be considered illegal.

Case 1143: Inconsistent Data Type Declarations

Gedae aborts when the same stream type is multiply defined. For example, suppose two primitives define data types with the same name but different definitions. If the primitives use these types in their `Input`, `Local` or `Output` sections, then Gedae aborts.

Case 1144: Function `appFree` Memory Leak

A command program running on VxWorks does not free all the resources allocated (memory, sockets, etc). The `appFree` function must release everything allocated. Gedae should automatically generate a call to `appFree` for the standard `exec-host` command program.

Case 1145: External Code Does Not Recompile

`Make` is not called after a successful run, so code changes to code listed in the `Personal_Emb_Obj_List` do not get recompiled. To force the recompile, it is currently necessary to change something from the Gedae GUI - like saving a primitive or toggling the Group "Run On Embedded" toggle off and on.

Case 1146: Large Graphs Fail to Display on Flattened Graph

If a graph is too large, then it cannot be displayed on the flattened graph. That is, if the flattened width or height exceeds the allowable pixmap width or height.

Case 1147: Primitive Cannot Recompile

If a primitive `Input`, `Output` or `Local` section is modified at runtime, then Gedae segfaults when the primitive is recompiled, and the graph is rerun. The user must currently exit Gedae after a primitive `Input`, `Output` or `Local` section has been modified.

Case 1148: VxWorks `embWallclock` Function Misses Wrap

When collecting trace information the `embWallclock` function timer can wrap without being detected. This failure to detect the timer wrap causes VxWorks processor timelines to appear compressed.

Case 1149: DSA with FanOut Does Not Work for Some BSPs

If one box output fans out to several boxes mapped to several different processors, then the DSA communication mechanism does not work correctly for Mercury and Sky BSPs.

Case 1150: Trace Table Saved on NT Not Readable on Solaris

Files saved from NT are byte reversed from what is expected on Solaris. Files saved on a big-endian platform cannot be read by Gedae from a little-endian platform.

Case 1151: FFT Primitives Only Work with Power of 2 Sized Vectors

The FFT boxes do not support non-power-of-2 lengths; however, the comments make no mention of this fact. If these boxes only support a power-of-2, then it would be useful to have a separate set of boxes that support a non-power-of-2.

Case 1152: Queues Don't Auto-resize When Opening Group Settings

If a queue needs to be autoresized and the `autoresize` option is turned on in the group-setting file, then the `autoresize` does not occur. A workaround for this problem is to toggle the Group Control Dialog `Autoresize` toggle off, and then on, and then resave the group settings.

Case 1153: Cannot Put Parentheses Inside Single Quotes

A line like `c = '{'` inside a primitive causes an error during primitive parsing. The parser will count the `'}` as a bracket and fail to find a matching closing bracket. Because brackets inside double quotes are ignored, the expression `"{"` will not cause a problem.

Case 1154: Set Partition by Equation Error

When setting a partition by equation on a subgraph family, and if any of the contained boxes are also families, then the `$1` variable applies to the deepest family member giving the wrong results.

Case 1155: Constants Propagated Through typedef Boxes

Constants propagated through typedef boxes cannot be used for instantiation

Case 1156: Partially Connected State Variable

State variables should not have to be connected to a primitive in every branch of an exclusive output. Currently, they must be connected when the branches are distributed.

Case 1157: Segmented Schedules with `nondet` Inputs

If a segmented static schedule has only `nondet` input queues, then it will not respond to segment ends correctly. This behavior was originally an intended feature of segmentation but needs to be either enforced during development or changed.

Case 1158: Primitives with `EndOfSegment` and No `Apply`

Primitives that have an `EndOfSegment` method but don't have an `Apply` method do not get included; therefore, the `EndOfSegment` method does not run. A workaround for this problem is to include an empty `Apply` method in the primitive.

Case 1159: Stream Box with `push` in Hostless Launch Package

If a stream box contains a call to `push` and it is made part of a hostless launch package, then the launch package will fail to compile, as the code for the `push` is not included in the standalone library.

Case 1161: `avail` on Variable Vectors is Unreliable

On variable vectors the `avail` function only returns the number of tokens available on the vector and not on the variable part. Usually these are the same, but when the variable vector is coming from another processor over a dynamic queue they may be updated at different times and may be different. As a result, `avail` may return a number too large. As a result when the data is processed, the variable part may not actually be ready, causing a segfault. A workaround is to check the `queues_ready` flag after the call to `amount` and only continue with execution if the `queues_ready` flag is non-zero.