Chapter 15. CG56 Domain

Authors: *Joseph T. Buck*
José Luis Pino

Other Contributors: *Brian L. Evans*
Chih-Tsung Huang
Christopher Hylands
Kennard White

15.1 Introduction

The CG56 domain generates assembly code for the Motorola 56000 series of digital signal processors. The graphs that we can describe in this domain follow the synchronous dataflow (SDF) model of computation. SDF allows us to schedule the `Blocks` and allocate all the resources at compile time. Refer to chapter "SDF Domain" on page 5-1 for a detailed description on the properties of SDF.

The Motorola 56000 series are fixed-point digital signal processors. The 56000 and 56001 processors have 24-bit data and instructions, and operate at a maximum clock rate of 40 MIPS. The 56100 processor has 16-bit data and instructions, operates at a maximum rate of 30 MIPS, and has analog/digital and digital/analog converters integrated on the chip. The 56301 has 24-bit data and instructions, operates at a maximum rate of 80 MIPS, and has several built-in input/output interfaces. Although the processors have pipelines of different lengths, the assembly code is backward compatible. The CG56 domain generates assembly code for the 56000 processor and has been tested on the Motorola simulator and on a 56001 board.

Since the 56000 processors are fixed point, the floating point data type has no meaning in the CG56 domain. Fixed-point values can take on the range [-1,1). The most positive value is $1 - 2^{-23}$ for the 56000 and 56300, and $1 - 2^{-15}$ for the 56100. The domain defines a new constant `ONE` set to this maximum positive value. In this chapter, whenever data types are not mentioned, fixed-point is meant. The complex data type means a pair of fixed-point numbers. The complex data type is only partially supported in that it is not supported for stars that have `anytype` inputs or outputs, except for `fork` stars. Integers are the same length as the fixed-point representation. Matrix data types are not supported yet.

Some of the demos use the Motorola 56000 assembler and simulator. You do not need to have a 56000 chip to run the simulator demos, the assemberl and simulator are available for downloading from Motorola at `http://www.mot.com/SPS/DSP/developers/clas.html`.

15.2 An overview of CG56 stars

The "open-palette" command in pigi (`O`) will open a checkbox window that you can use to open the standard palettes in all of the installed domains. For the CG56 domain, the star library is large enough that it has been divided into sub-palettes as was done with the SDF main palette.

The top-level palette is shown in figure 15-1. The palettes are Signal Sources, I/O, Arithmetic, Nonlinear Functions, Logic, Control, Conversion, Signal Processing, and Higher Order Functions. The stars on the Higher Order Functions (HOF) palette are used to help lay out schematics graphically. The HOF stars are in the HOF domain, and not the CG56 domain. The names of the others palettes are modeled after the SDF star palettes of the same name in section 5.2 on page 5-4, except the I/O palette which contains target-specific I/O stars for the Ariel S-56X DSP board and the Motorola 56001 simulator. Each palette is summarized in more detail below. More information about each star can be obtained using the on-line “profile” command (`,`), the on-line man command (`M`), or by looking in the *Star Atlas* volume of *The Almagest*.

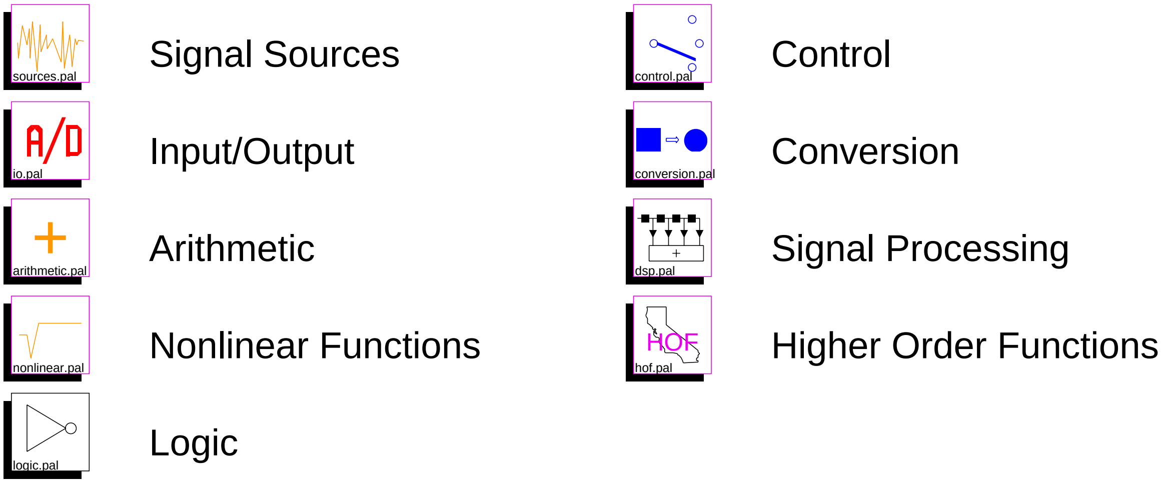

FIGURE 15-1: The palette of star palettes for the CG56 domain.

At the top of each palette, for convenience, are instances of the delay icon, the bus icon, and the following star:

`BlackHole` Discard all inputs. This star is useful for discarding signals that are not useful.

15.2.1 Source stars

Source stars are stars with only outputs. They generate signals, and may represent external inputs to the system, constant data, or synthesized stimuli. The palette of source stars is shown in figure 15-2. Refer to 5.2.1 on page 5-5 for descriptions of the SDF equivalent stars: `Const`, `ConstCx`, `ConstInt`, `Ramp`, `RampInt`, `Rect`, `singen`, and `WaveForm`.

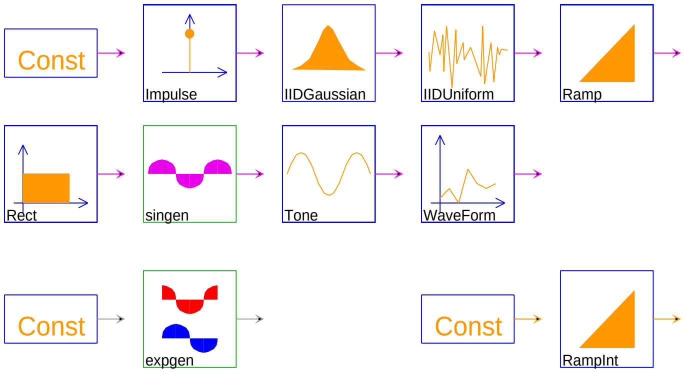

FIGURE 15-2: The palette of source stars for the CG56 domain.

`Impulse` Generate a single impulse of size *impulseSize* (default `ONE`).

`IIDGaussian` Generate a white Gaussian pseudo-random process with mean 0 and standard deviation 0.1. A Gaussian distribution is realized by summing *noUniforms (default* 16) number of uniform random variables. According to the central limit theorem, the sum of N random variables approaches a Gaussian distribution as N approaches infinity.

`IIDUniform` Generate an i.i.d. uniformly distributed pseudo-random process. Output is uniformly distributed between -*range* and *range* (default `ONE`).

`Tone` Generate a sine or cosine wave using a second order oscillator. The wave will be of *amplitude* (default 0.5), *frequency* (default 0.2), and *calcType* (default “sin”)

15.2.2 I/O Stars

I/O stars are target specific stars that allow input and output of stimuli to a target architecture. Currently there are I/O stars for both the Ariel S-56X DSP and the Motorola 56k simulator which are divided hierarchically as shown in figure 15-3.

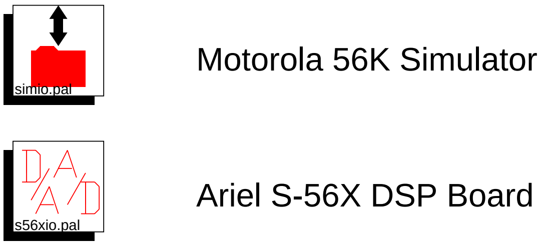

FIGURE 15-3: CG56 I/O Palette

Motorola 56000 Simulator I/O Stars

The palette of I/O stars for the Motorola 56K simulator target is shown in figure 15-4.

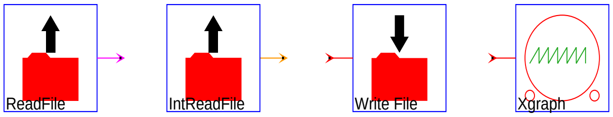

FIGURE 15-4: Motorola Simulator I/O Palette

`ReadFile` Read fixed-point ASCII data from a file. The simulation can be halted on end-of-file, or the file contents can be periodically repeated, or the file contents can be padded with zeros.

`IntReadFile` Read integer ASCII data from a file. The simulation can be halted on end-of-file, or the file contents can be periodically repeated, or the file contents can be padded with zeros.

`WriteFile` Write data to a file. The simulator dumps the data presented at the input of this star into a specified file.

`Xgraph` This star shares the same parameters as its SDF and CGC star equivalents. However, with this star, you can only have one input signal. See “pxgraph — The Plotting Program” on page 20-1 to learn about plotting options.

Ariel S-56X DSP Board I/O Stars

The s56xio palette (figure 15-5) allows I/O to the Ariel S-56X DSP board. To use these blocks, you will need access to a S-56X DSP board. These blocks are divided into three subcategories: generic S-56X, QDM S-56X and CGC-S56X. The QDM stars requires installing qdm, a debugger for DSP systems which was developed by Phil Lapsley at U.C. Berkeley. Qdm is currently available from Mike Peck[1], the designer of the S-56X board.

Generic S-56X

`adjustableGainGX` Create an interactive adjustable gain using `HostSliderGX`.

`da` Send the input to both input ports of the SSI star.

`HostAOut` Output data from the DSP to host via host port asynchronously.

`HostSldrGX` Generate an athena widget slider for interactive asynchronous input over the host port.

`MagnavoxIn` Read data from a Magnavox CD player.

`Magnavox` Read data from and write data to a Magnavox CD player.

`MagnavoxOut` Write data to a Magnavox CD player.

`PrPrtAD` Read from the A/D in Ariel ProPort.

`PrPrtADDA` Read from the A/D and write to the D/A on the Ariel ProPort.To use both the A/D and D/A on a ProPort you must use this star

[1] Mike Peck, Berkeley Camera Engineering, mpeck@bcam.com (http://www.bcam.com)

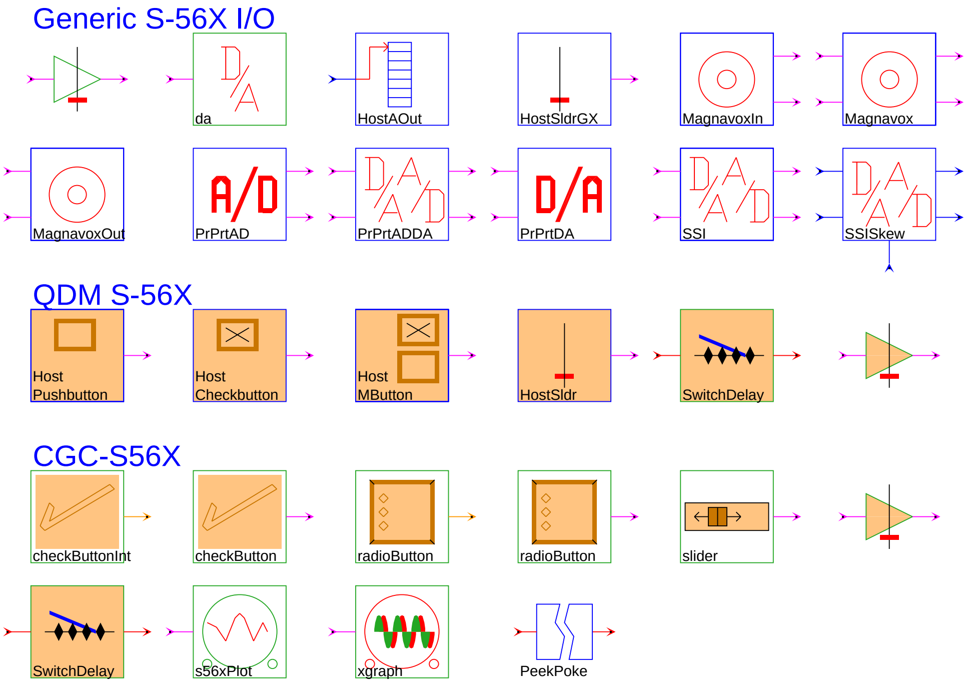

FIGURE 15-5: S-56X I/O Palette

and not the separate A/D and D/A stars.

`PrPrtDA` Write to the D/A on the Ariel ProPort.

`SSI` A generic input/output star for the DSP56001 SSI port.

`SSISkew` Interface to the 56001 SSI's port with timing-skew capability.

QDM S-56X

To use these stars you must have qdm installed and be using the uniprocessor s-56x target. The target parameter *monitor* must be set to `qdmterm_s56x -run`.

`HostButton` (2 icons) Graphical two-valued input source. There are two types of buttons: push-buttons and check-buttons. Both present a single button to the user that may be “pressed” with the mouse. The buttons differ in the semantics of the push. When the pushbutton is pressed, the *onVal* state is output, otherwise *offVal*.

`HostMButton` Graphical one-of-many input source. The star always outputs one of a finite number of values: the output is controlled by the user selecting one of several buttons. Exactly one button in the group is on.

`HostSldr` Graphical host slider for asynchronous input source.

`SwitchDelay` This galaxy synchronously switches between the input value and the value of the input delayed by TotalDelay (default 8000) samples.

`adjustableGain` A user adjustable gain, uses `HostSlider`.

CGC-S56X

`checkButtonInt` This galaxy creates a Tk checkbutton widget that produces the given *onValue* (default 1) when pressed and *offValue* (default 0) otherwise.

`checkButton` This galaxy creates a Tk checkbutton widget that produces the given *onValue* (default 1.0) when pressed and *offValue* (default 0.0) otherwise.

`radioButtonInt` This galaxy creates a Tk radiobutton widget that allows the user to select from among a set of possible output values given by pairs (default “One 1” “Two 2”).

`radioButton` This galaxy creates a Tk radiobutton widget that allows the user to select from among a set of possible output values given by pairs (default “One 1” “Two 2”)

`slider` This galaxy creates a Tk slider widget that produces the given value indicated by the slider position which is between low (default 0.0) and high (default 1.0) and initially set to value (default 0.0).

`adjustableGain` This galaxy multiplies the input by a gain value taken from a Tk slider position between low (default 0.0) and high (default 1.0), which is initially set to value (default 0.0).

`SwitchDelay` This galaxy synchronously switches between the input value and the value of the input delayed by *TotalDelay* (default 8000) samples.

`s56XPlot` This galaxy plots the input interactively using TkPlot.

`Xgraph` This galaxy simply contains a `CGCXgraph` star for use in a CG56 galaxy. The galaxy parameters are identical to those of the enclosed star.

`PeekPoke` Nondeterminate communication link that splices in a peek/poke pair. In this context, it provides a link between the S-56X Motorola 56001 board and the workstation.

15.2.3 Arithmetic stars

The arithmetic stars that are available are shown in figure 15-6.

`Add` (2 icons) Output the sum of the inputs. If *saturation* is set to yes, the output will saturate.

`Sub` Outputs the “pos” input minus all of the “neg” inputs.

`Mpy` (2 icons) Outputs the product of all of the inputs.

`Gain` The output is set the input multiplied by a *gain* term. The gain must be in [-1,1).

AddCx (2 icons) Output the complex sum of the inputs. If *saturation* is set to yes, the output will saturate.

SubCx Outputs the “pos” input minus all of the “neg” inputs.

MpyCx (2 icons) Outputs the product of all of the inputs.

AddInt (2 icons) Output the sum of the inputs. If *saturation* is set to yes, the output will saturate.

SubInt Outputs the “pos” input minus all of the “neg” inputs.

MpyInt (2 icons) Outputs the product of all of the inputs.

GainInt The output is set the input multiplied by an integer *gain* term.

DivByInt This is an amplifier. The integer output is the integer input divided by the integer *divisor* (default 2). Truncated integer division is used.

MpyRx Multiply any number of rectangular complex inputs, producing an output.

MpyShift Multiply and shift.

Neg Output the negation of the input.

Shifter Scale by shifting left *leftShifts* bits. Negative values of *leftShifts* implies right shifting.

15.2.4 Nonlinear stars

The nonlinear palette (figure 15-7) in the CG56 domain includes transcendental functions, quantizers, table lookup stars, and miscellaneous nonlinear functions.

Abs Output the absolute value of the input.

ACos Output the inverse cosine of the input, which is in the range -1.0 to 1.0. The output, in the principle range of 0 to π, is scaled down by π.

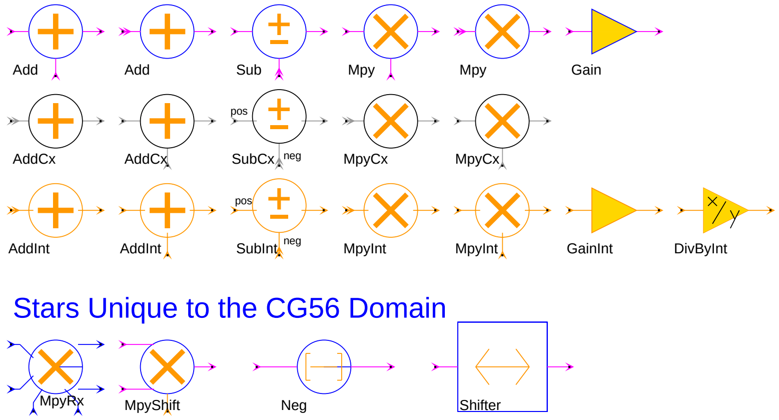

FIGURE 15-6: CG56 Arithmetic Palette

`ASin` Output the inverse sine of the input, which is in the range -1.0 to 1.0. The output, in the principle range of $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, is scaled down by π.

`Cos` Output the cosine, calculated the table lookup. The input range is [-1,1] scaled by π.

`expjx` Output the complex exponential of the input.

`Intgrtr` An integrator with leakage set by *feedbackGain*. If there is an overflow, the *onOverflow* parameter will designate a wrap around, saturate or reset operation.

`Limit` Limits the input between the range of [*bottom*, *top*].

`Log` Outputs the base two logarithm.

`MaxMin` Output the maximal or minimal (*MAX*) sample out of the last *N* input samples. This can either *compareMagnitude* or take into account the sign. If *outputMagnitude* is `YES` the magnitude of the result is written to the output, otherwise the result itself is written.

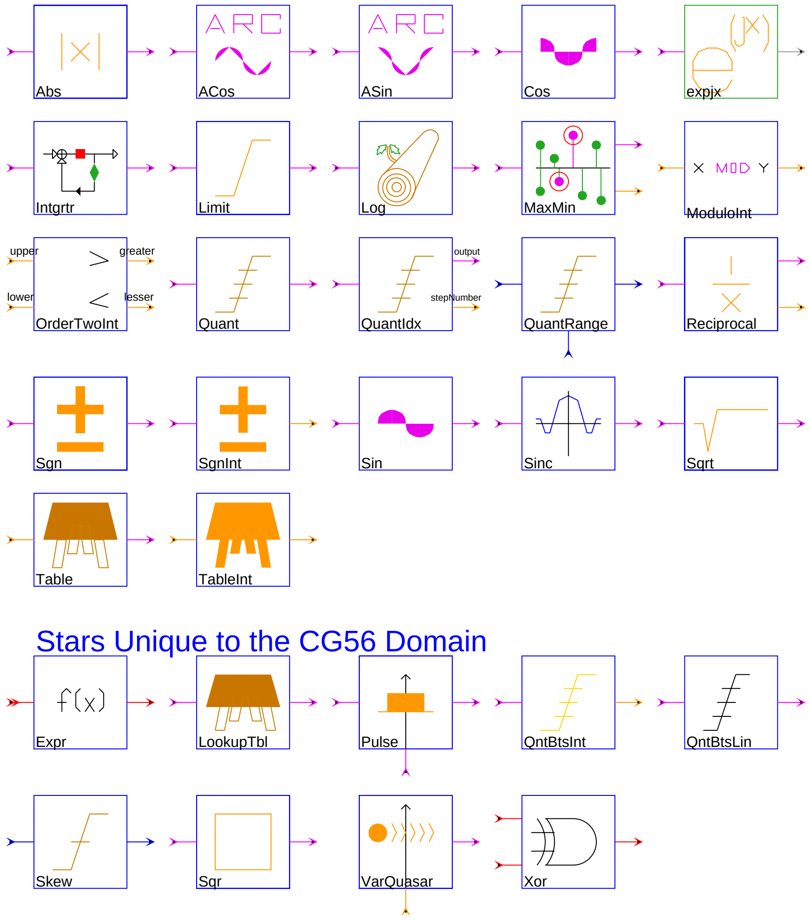

FIGURE 15-7: CG56 Nonlinear Palette

`ModuloInt` Output the remainder after dividing the integer input by the integer *modulo* parameter.

`OrderTwoInt` Takes two inputs and outputs the greater and lesser of the two integers.

`Quant` Quantizes the input to one of N+1 possible output *levels* using N *thresholds*.

`QuantIdx` The star quantizes the input to one of N+1 possible output *levels* using N *thresholds*. It also outputs the index of the quantization level used.

`QuantRange` Quantizes the input to one of N+1 possible output *levels* using N *thresholds*.

`Reciprocal` Outputs the reciprocal to *Nf* precision in terms of a fraction and some left shifts.

`Sgn` Outputs the sign of the input.

`SgnInt` Outputs the sign of the integer input.

`Sin` Outputs the sine, calculated using a table lookup. The input range is [-1,1) scaled by π.

`Sinc` Outputs the sinc functions calculated as sin(x)/x.

`Sqrt` Outputs the square root of the input.

`Table` Implements a real-valued lookup table. The *values* state contains the values to output; its first element is element zero. An error occurs if an out of bounds value is received.

`TableInt` Implements an integer-valued lookup table. The *values* state contains the values to output; its first element is element zero. An error occurs if an out of bounds value is received.

`Expr` General expression evaluation.

`LookupTbl` The input accesses a lookup table. The *interpolation* parameter determines the output for input values between table-entry points. If *interpolation* is “linear” the star will interpolate between table entries; if *interpolation* is set to “none”, it will use the next lowest entry.

`Pulse` Generates a variable length pulse. A pulse begins when a nonzero trigger is received. The pulse duration varies between 1 and *maxDuration* as the control varies between [-1,1).

`QntBtsInt` Outputs the two’s complement number given by the top *noBits* of the input (for integer output).

`QntBtsLin` Outputs the two’s complement number given by the top *noBits* of the input, but an optional *offset* can be added to shift the output levels up or down.

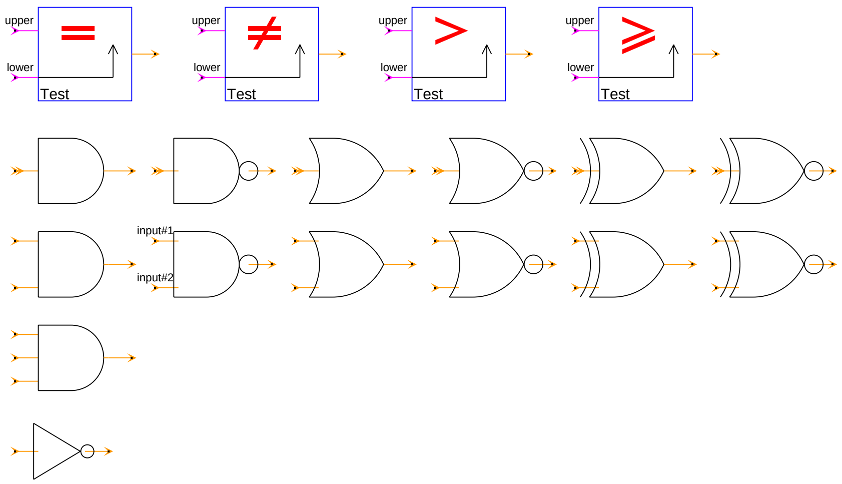

FIGURE 15-8: CG56 Logic Palette

`Skew` Generic skewing star.

`Sqr` Outputs the square of the input.

`VarQuasar` A sequence of values(*data*) is repeated at the output with period N (integer input), zero-padding or truncating the sequence to N if necessary. A value of O for N yields an aperiodic sequence.

`Xor` Output the bit-wise exclusive-or of the inputs.

15.2.5 Logic stars

The Logic stars are discussed below:

`Test` (4 icons) Test to see if two inputs are equal, not equal, greater than, and greater than or equal. For less than and less than or equal, switch the order of the inputs.

`And` (3 icons) True if all inputs are non-zero.

`Nand` (2 icons) True if all inputs are not non-zero.

`Or` (2 icons) True if any input is non-zero.

`Nor` (2 icons) True if any input is zero.

`Xor` (2 icons) True if an odd number of inputs is non-zero.

`Xnor` (2 icons) True if an even number of inputs is not non-zero.

`Not` Logical inverter.

15.2.6 Control stars

Control stars (figure 15-9) manipulate the flow of tokens. All of these stars are polymorphic; they operate on any data type. Refer to 5.2.6 on page 5-17 for descriptions of the SDF equivalent stars: `Fork`, `DownSample`, `Commutator`, `Distributor`, `Mux`, `Repeat`, `Reverse`, and `UpSample`.

`ChopVarOffset` This star has the same functionality as the `Chop` star except now the *offset* parameter is determined at run time through a control input.

`Cut` On each execution, this star reads a block of *nread* samples (default 128) and writes *nwrite* of these samples (default 64), skipping the first offset samples (default 0). It is an error if *nwrite* + *offset* > *nread*. If *nwrite* > *nread*, then the output consists of overlapping windows, and hence *offset* must be negative.

`Delay` A delay star of parameter *totalDelay* unit delays.

`Pad` On each execution, Pad reads a block of *nread* samples and writes a block of *nwrite* samples. The first *offset* samples have value *fill*, the next *nread* output samples have values taken from the inputs, and the last *nwrite* - *nread* - *offset* samples have value *fill* again.

`Rotate` The star reads in an input block of a certain *length* and performs

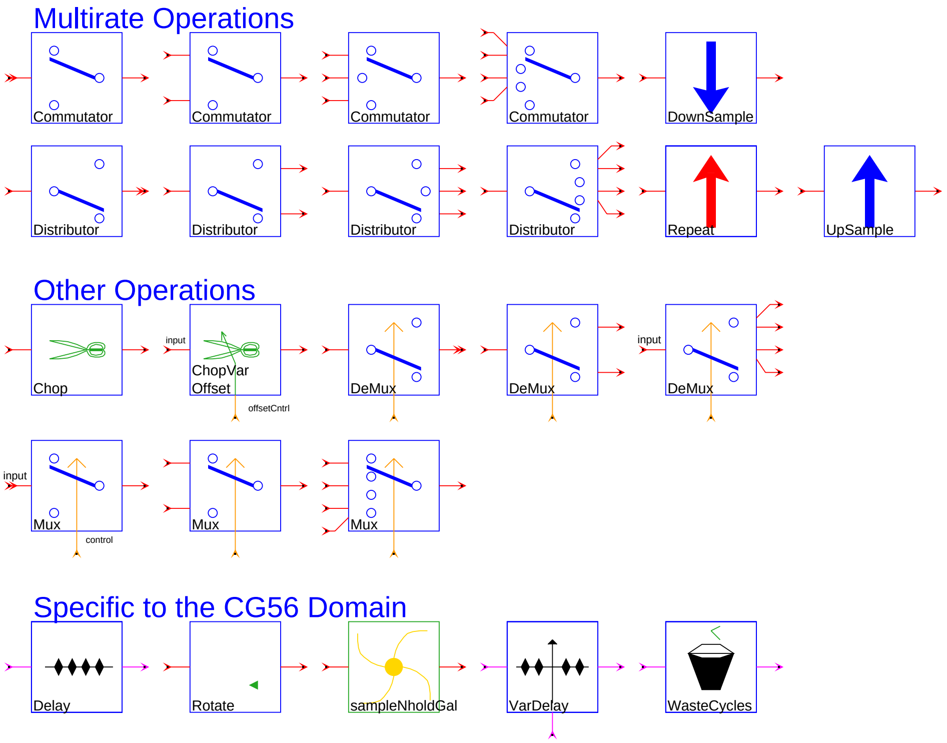

FIGURE 15-9: CG56 Control Palette

a circular shift of the input. If the *rotation* is positive, the input is shifted to the left so that ouput[0] = input[*rotation*]. If the *rotation* is negative, the input is shifted to the right so that output[*rotation*] = input[0].

`sampleNholdGalaxy`
This sample-and-hold galaxy is more memory efficient than using a downsample star for the same purpose.

`VarDelay` A variable delay that will vary between 0 and *maxDelay* as the control input varies between -1.0 and 1.0.

`WasteCycles` Stalls the flow of data for *cyclesToWaste* number of cycles.

15.2.7 Conversion stars

The palette in figure 15-10 shows stars for format conversions from fixed point to complex fixed point. The complex data type is only partially implemented in CG56. Complex ports can be connected only to complex ports. Anytype ports can only be connected to fixed and integer ports

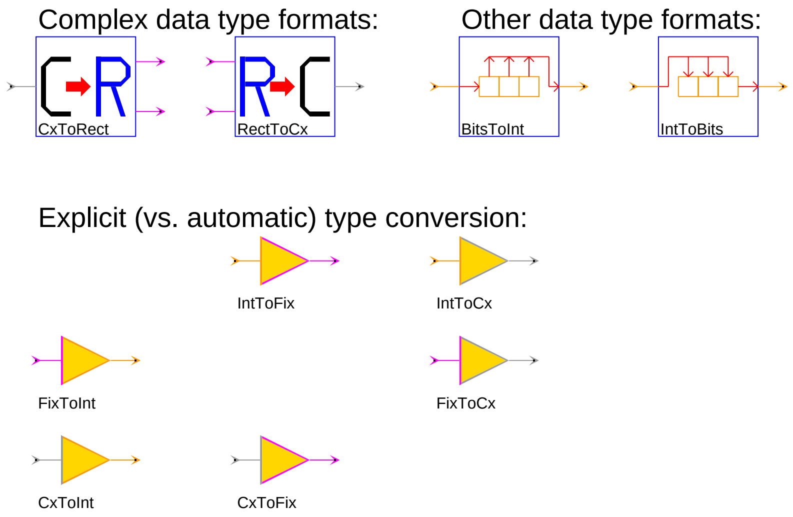

FIGURE 15-10: CG56 Conversion Palette

`CxToRect` Output the real part and imaginary part of the input of separate output ports.

`RectToCx` Output a complex signal with real and imaginary part inputs.

`BitsToInt` Convert a stream of bits to an integer.

`IntToBits` Convert an integer into a stream of bits.

`FixToCx` Convert fixed-point numbers to complex fixed-point numbers.

`FixToInt` Convert fixed-point numbers to complex fixed-point numbers.

`CxToFix` Convert fixed-point numbers to complex fixed-point numbers.

`CxToInt` Convert fixed-point numbers to complex fixed-point numbers.

`IntToFix` Convert fixed-point numbers to complex fixed-point numbers.

`IntToCx` Convert fixed-point numbers to complex fixed-point numbers.

15.2.8 Signal processing stars

The palette shown in figure 15-11 has icons for the library of signal processing functions. The filter stars follow. The `Goertzel` and `IIR` stars are identical to their SDF counterparts.

`Allpass` An allpass filter with one pole and one zero. The location of these is given by the "polezero" input.

`Biquad` A two-pole, two-zero IIR filter (a biquad).

$$H(z) = \frac{1 + n_1 z^{-1} + n_2 z^{-2}}{1 + d_1 z^{-1} + d_2 z^{-2}}$$

`Comb` A comb filter with a one-pole lowpass filter in the delay loop.

`BiquadDSPlay` A two-pole, two zero IIR filter (a biquad). This biquad is tailored to use the coefficients from the DSPlay filter design tool. If DSPlay gives the coefficients: A B C D E then define the parameters as follows: a=A, b=B, c=C, d=-(D+1), e = -E. This

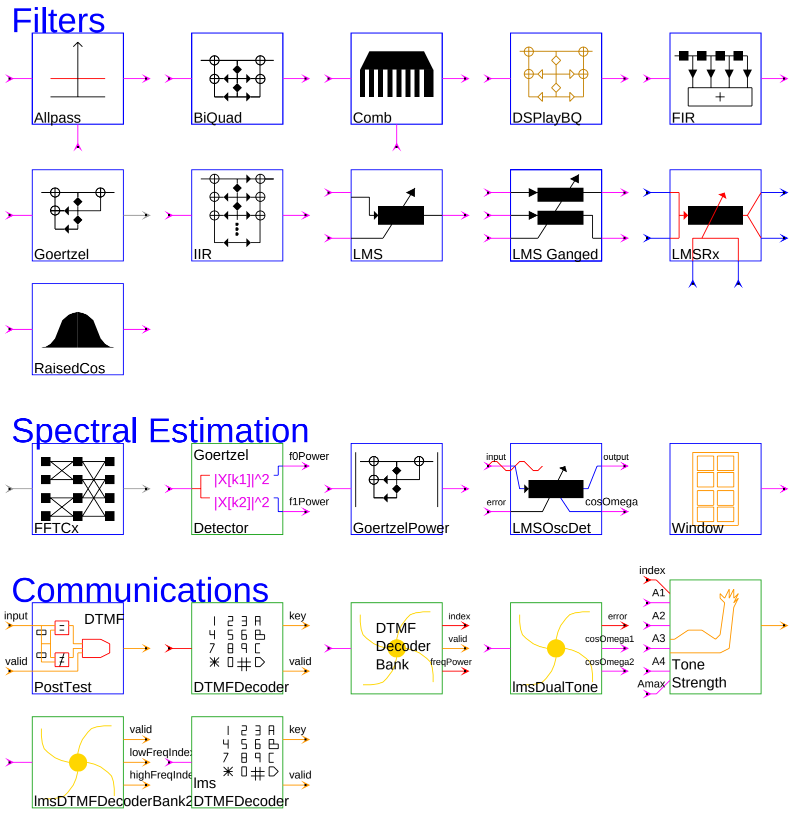

FIGURE 15-11: CG56 Signal processing Palette

only works if a, b, c, d, and e, are in the range [-1,1). The default coefficients implement a low pass filter.

$$H(z) = \frac{a + bz^{-1} + cz^{-2}}{1 - (d + 1)z^{-1} - ez^{-2}}$$

`FIR` A finite impulse response (FIR) filter. Coefficients are specified by the *taps* parameter. The default coefficients give an 8th order, linear-phase, lowpass filter. To read coefficients from a file, replace the default coefficients with `< filename`, preferably specifying a complete path. Polyphase multirate filtering is also supported.

`LMS` An adaptive filter using the LMS adaptation algorithm. The initial coefficients are given by the *coef* parameter. The default initial coefficients give an 8th order, linear phase lowpass filter. To read default coefficients from a file, replace the default coefficients with `< filename`, preferably specifying a complete path. This star supports decimation, but not interpolation.

`LMSGanged` A LMS filter were the coefficients from the adaptive filter are used to run a FIR filter in parallel. The initial coefficients default to a lowpass filter of order 8.

`LMSRx` A Complex LMS filter

`RaisedCos` An FIR filter with a magnitude frequency response shaped like the standard raised cosine used in digital communications. See the `SDFRaisedCosine` star for more information.

The spectral estimation stars follow. The `GoertzelDetector`, `GoertzelPower`, and `LMSOscDet` are identical to their SDF counterparts.

`FFTCx` Compute the discrete-time Fourier transform of a complex input using the fast Fourier transform (FFT) algorithm. The parameter *order* (default 8) is the transform size. The parameter *direction* (default 1) is 1 for forward, -1 for the inverse FFT.

`Window` Generate standard window functions or periodic repetitions of standard window functions. The possible functions are `Rectangle`, `Bartlett`, `Hanning`, `Hamming`, `Blackman`, `SteepBlackman`, and `Kaiser`. One period of samples is produced on each firing.

The communications stars are exactly like their SDF counterparts.

15.3 An overview of CG56 Demos

A set of CG56 demonstration programs have been developed. A top-level palette, shown in figure 15-12, contains an icon for each demo palette. The demos are grouped by the CG56 target on which they are implemented. If you do not have the require compiler, simulator, or DSP card, then you can still run the demos to see the generated code. To do this make sure that the *run* and *compile* target parameters are to `NO`. By default, the generated code is written to `$HOME/PTOLEMY_SYSTEMS` directory.

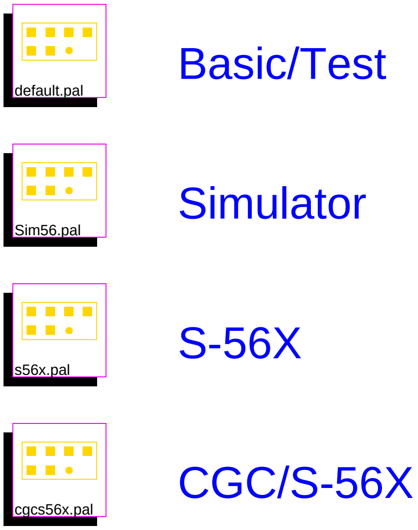

FIGURE 15-12: The top-level demo palette for the CG56

15.3.1 Basic/Test demos

The Basic/Test palette contains six demonstrations (figure 15-13).

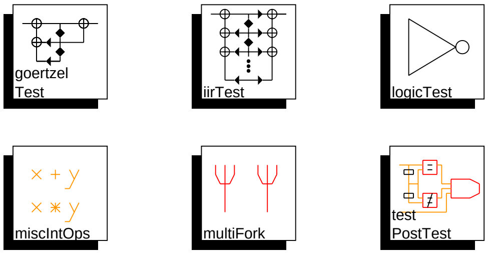

FIGURE 15-13: Basic Demo Palette

`goertzelTest` Test the Goertzel filters for computing the discrete Fourier transform.

`iirTest` Test the infinite impulse response (IIR) filters.

`logicTest` Test various comparison tests and Boolean functions.

`miscIntOps` Test integer arithmetic operations.

`multiFork` Test the `AnyAsmFork` star. An `AnyAsmFork` star is one of a group of stars that do produce any code at compile time.

`testPostTest` Test the `DTMFPostTest` star used in touchtone decoding.

15.3.2 Motorola Simulator Demos

The demos in palette figure 15-14 will generate stand alone applications. These applications will consist of: a shell script to control the simulator and output display programs; a simulator command file; and the assembled code to run on the simulator. The simulator can be run in either an interactive mode or in the background by setting the *interactive* target parameter.

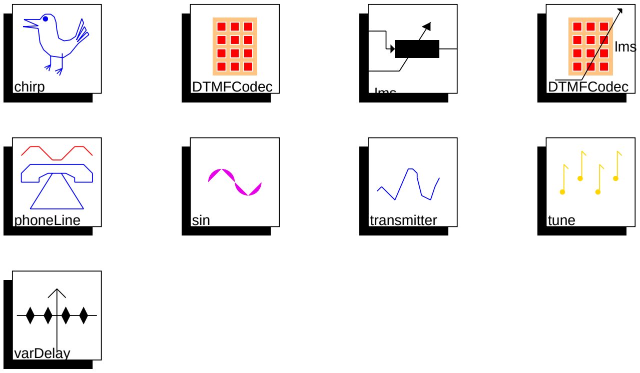

FIGURE 15-14: Motorola Simulator Demos

`chirp` This system uses two integrators and a cosine to generate a chirp signal.

`DTMFCodec` Demonstration of touchtone detection using the discrete Fourier transform implemented by using Goertzel filters.

`lms` A noise source is connected to an eighth-order least-mean squares (LMS) adaptive filter with initial taps specifying a low-pass filter. The taps adapt to a null filter (the impulse response is an impulse) and the error signal is displayed.

`lmsDTMFCodec` Demonstration of touchtone detection using Normalized Direct Frequency Estimation implemented by using Least-Mean Squares (LMS) adaptive filters.

`phoneLine` A telephone channel simulator. A tone is passed through some processing which implements various distortions on a telephone channel. The parameters that are controllable are: noise, channel filter, second harmonic, third harmonic, frequency offset, phase jitter frequency, and phase jitter amplitude.

`sin` A sine wave is generated by using two integrators in a feedback loop.

`transmitter` A simple 4-level PAM transmitter

`tune` A tune is generate using FM synthesis of notes stored in a table. The sounds produced are not particularly musically appealing, partly because the modulation index is not variable and the attack and decay profiles are too limited.

`varDelay` This is a simple application demonstrating variable delay with linear interpolation.

15.3.3 S-56X Demos

The demos shown in figure 15-15 require an Ariel S-56X DSP board to be installed in the workstation. In addition, all but the first demo requires QDM. These demos generate a stand alone application consisting of: a shell script to download and run the assembled code; a file specifying the asynchronous user I/O interface; and the assembled code.

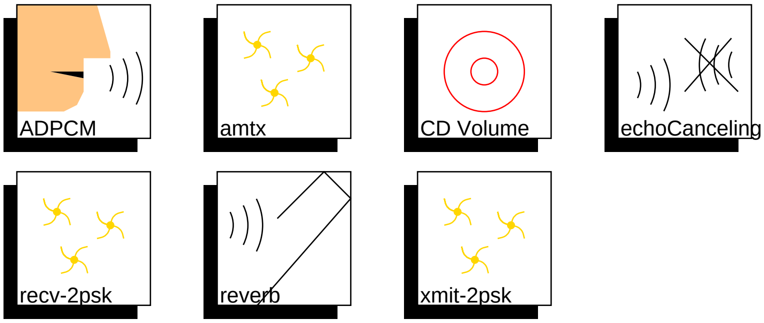

FIGURE 15-15: Ariel S-56X DSP Board demos

`ADPCM` This demo implements a ADPCM coder and decoder. The user at run time can vary the number of quantization bits, the quantization range, and a delay so that signal can be heard instantaneously or a second later. Requires an Ariel Proport and a microphone.

`amtx` Amplitude Modulation Transmitter. The results of the transmitter are displayed asynchronously at run time.

`CD Volume` A universe showing a implementing a volume control with `CG56HostSliderGX` stars. Requires a modified CD player.

`echoCanceling` A system implementing a pair of echo cancellation filters. The first echo cancellation filter cancels an artificial echo introduced by an FIR filter. The second echo cancellation filter is used to cancel the echoes produced by have one microphone next to loud speaker. Another microphone is used for desired input, such as speech. Requires an Ariel Proport and two microphones.

`recv-2psk` 2-PSK Bandpass filter.

`reverb` This system implements a reverberation system using Comb filters. Requires an Ariel Proport and a microphone.

`xmit-2psk` 2-PSK transmitter.

15.3.4 CGC-S56X Demos

All of the demos in this palette use the `CompileCGSubsystems` target described in section 13.4 on page 13-10.

Stand alone Application Demos

The first set demos generate stand alone applications consisting of two parts: a program generate in C that implements the sub-graph that runs on the host, and a program generated in Motorola 56k assembly that is to be run on the S-56X. The C program initializes and downloads the S-56X program automatically. The first two of the demos shown in figure 15-16, `lms`, `phoneLine`, `DTMFCodec` and `lmsDTMFCodec` are identical to the simulator demos.

`Modem` The modem palette contain 3 phased shift keying modem demos. These demos illustrate the use of peek/poke actors and hierarchical scheduling. Requires an Ariel Proport and a microphone.

`dtmfSpectrum` This demos implements a DTMF tone generator and displays the resultant frequency spectrum.

`synth` A FM music synthesis demonstration. Requires an Ariel Proport.

`synthFFT` A FM music synthesis demonstration showing the resultant frequency spectrum. Requires an Ariel Proport.

`PRfilterBank` A perfect reconstruction filter bank.

`ADPCM` This demo implements a ADPCM coder and decoder. The user at run time can vary the number of quantization bits, the quantization range, and a delay so that signal can be heard instantaneously or a second later. Requires an Ariel Proport and a microphone.

Simulation SDF-Wormhole Demos

The simulation SDF wormhole demos create simulation SDF stars in ptlang and also a load file for the S-56X card. Unlike the other CG56 demos, the applications produced here will not run as stand alone applications. The wormhole allows the user to imbed a CG56 sys-

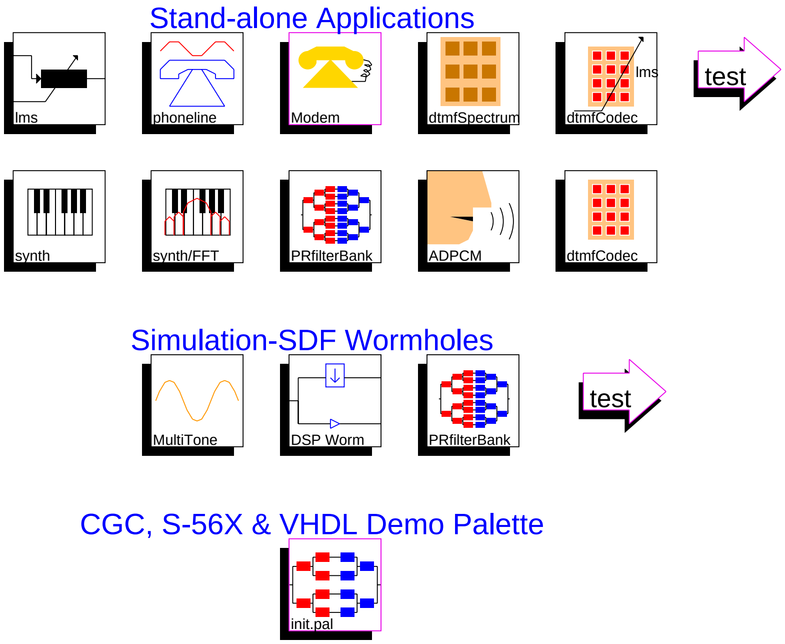

FIGURE 15-16: CGC S-56X demos

tem running on a Ariel S-56X DSP board into a Ptolemy simulation.

`MultiTone` Generates three sine waves on the S-56X which are at different rates relative to one another.

`DSPWorm` Demonstrates multirate I/O between Ptolemy and the S-56X board.

`PRfilterBank` A perfect reconstruction filter bank.

CGC, S-56X & VHDL Demos

The demos in this palette all implement some for of a perfect reconstruction filter bank. One of the examples generates a simulation SDF star which makes use of a VHDL simulator, the S-56X DSP card and the workstation.

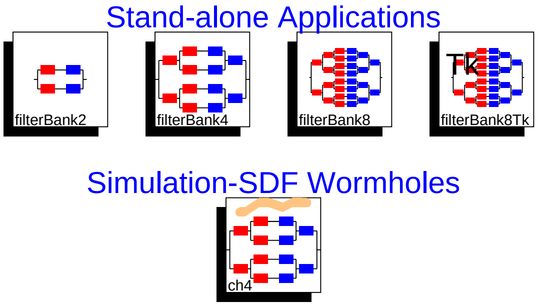

FIGURE 15-17: Combined CGC, CG56, VHDL demos

15.4 Targets

Seven CG56 targets are included in the Ptolemy distribution. To choose one of these targets, with your mouse cursor in a schematic window, execute the Edit:edit-target command (or just type “T”). You will get a list of the available `Targets` in the CG56 domain. The `default-CG56` target is the default value. When you click `OK`, the dialog box appears with the parameters of the target. You can edit these, or accept the defaults. The next time you run the schematic, the selected target will be used.

15.4.1 Default CG56 (default-CG56) target

The default target is used only for code generation. It has the following set of options:

host (`STRING`) Default =
The default is the empty string. Host machine to compile or assemble code on. All code is written to and compiled and run on the computer specified by this parameter. If a remote computer is specified here then `rsh` commands are used to place files on that computer and to invoke the compiler. You should verify that your .rhosts file is properly configured so that `rsh` will work.

directory (`STRING`) Default = `$HOME/PTOLEMY_SYSTEMS`
This is the directory to which all generated files will be written to.

file (`STRING`) Default =
The default is the empty string. This represents the prefix for file names for all generated files.

Looping Level Specifies if the loop scheduler should be used. Please refer to the section “default-CG” on page 13-2 for more details on this option. Refer to “Default SDF target” on page 5-65 and “The loop-SDF target” on page 5-67 for more details on loop scheduling.

display? (`INT`) Default = `YES`
If this flag is set to `YES`, then the generated code will be displayed on the screen.

compile? This is a dummy flag since the default target only generates code.

run? This is a dummy flag since the default target only generates code.

xMemMap (`STRING`) Default = `0-4095`
Valid x memory address locations. Default is `0-4095`, which means x:0 through x:4095 are valid memory addresses. Disjoint segments of memory can be specified by separating the contiguous ranges with spaces, e.g. “0-4095 5000-5500.”

yMemMap (`STRING`) Default = `0-4095`
Valid y memory address locations. Default is `0-4095`, which means y:0 through y:4095 are valid memory addresses.

subroutines? (`INT`) Default = `-1`
Setting this parameter to N makes the target attempt to generate a subroutine instead of in-line code for a star if the number of repetitions of that star is greater than N (use N=0 to generate subroutines even for stars with just 1 repetition). Set “*subroutines?*” to `-1` (or any other negative integer) to disable the feature.

show memory usage? (`INT`) Default = `NO`
If `YES`, then the target will report the actual amount of program, X data memory, and Y data memory used by the program in words.

15.4.2 CG56 Simulator (sim-CG56) target

This target is used for generating DSP56000 assembly code, assembling it, and running it on a Motorola DSP56000 simulator. For this to work properly, the Motorola 56000 assembler (asm56000) and the simulator (sim56000) must be in the user path. Otherwise a run on this target produces code only, and an error message will appear indicating the absence of the required programs in the user path. Input and output files specified in `ReadFile` and `WriteFile` stars are passed on to the simulator by an automatically generated `universe.cmd` file, which is sourced by the simulator.

The options for this target are mostly the same as the ones for `default-CG56` above, except for the following:

compile? (`INT`) Default = `YES`

If this option is set to `YES`, then generated code is assembled using `asm56000` program.

run? (`INT`) Default = `YES`
If `YES`, then the assembled code is run on the Motorola simulator sim56000.

Interactive Sim. (`INT`) Default = `YES`
If `YES` the simulator is run interactively (in which case one can add breakpoints, single step through code, etc.)

15.4.3 Ariel S-56X (S-56X) target

This target generates stand alone applications that will run on the Ariel S-56X DSP board. An optional graphical debugger, QDM, is available from the board designer, Mike Peck. This debugger is needed for some of the user I/O stars that are specific to this target.

The options for this target are mostly the same as the ones for `default-CG56`, except for the following:

monitor (`STRING`) Default =
The default is the empty string.This parameter specifies an optional monitor of debugger for use with the S-56X target. If the application has QDM stars, this parameter should be set to `qdmterm_s56x -run`.

15.4.4 CG56 Subroutine (sub-CG56) target

This target is used to generate subroutines that can be called from hand-written 56000 code. The options are identical to those of `default-CG56` target.

15.4.5 Multiprocessor 56k Simulator (MultiSim-56000) target

This target generates code for a multiprocessor DSP system, where the processors communicate via shared memory. Unfortunately the multiprocessor simulator is not available outside of U.C. Berkeley.

The options for this target are mostly the same as the for `CGMultiTarget`, except for the following:

sMemMap (`STRING`) Default = `4096-4195`
Specifies the shared memory map to use for the communication stars.