filtorX: computer-aided filter design and industry

C. Ouslis, A.S. Sedra, W.M. Snelgrove?, S. Good*

Dept. of Elec. Eng., University of Toronto, Toronto, Ontario, M5S lA4,
e-mail: ouslis@eecg.toronto.edu fax:(416)978-7423

Dept. of Electronics, Carleton University, Ottawa, Ontario, KlS 5B6
Mitel Semiconductor Corp., Kanata, Ontario, K2K 1x5

Abstract-Switched-capacitor transmit and receive filters for a CCITT compliant codec were designed at Mite1 Semiconductor employing hand-coded Pascal[12] programs and a variety of other tools. This approach required a significant investment in the designer's time: to manually optimise biquadratic filter section performance; to write and debug the Pascal programs used and to generate simulation netlists. filtorX, written at the University of Toronto, is a user-programmable, computer-aided filter design tool intended for research and industrial design. It enables designers to perform diverse functions automatically, using one environment from which external simulation tools can be operated and is applied to the challenges of this project.

1 Problem Definition

CCITT compliant transmit and receive filters for integration into a telecommunications PCM-codec chip were required. CMOS technology was to be used for voice-band frequencies, leading to switched-capacitor circuitry as the obvious choice. The transmit filter, shown in Fig. 1, required 1) low-frequency rejection to suppress line noise from AC power sources and 2) attenuation of signals above 3.4kHz as an anti-aliasing filter for the PCM encoder. The receive filter, an anti-imaging filter which follows the D/A converter, has the primary function of

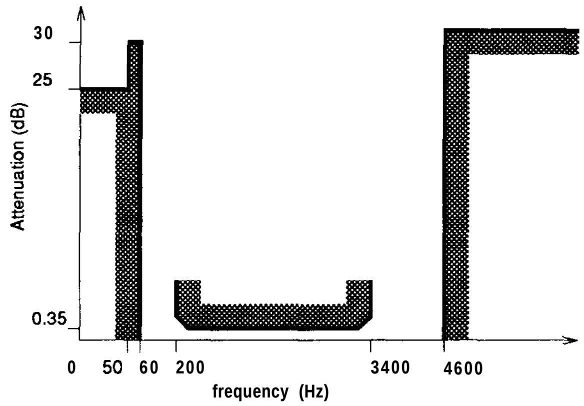


Fig.1. The CCITT transmit specification.

removing images above 3.4kHz and has no stringent requirements for low-frequency signals. This specification is plotted in Fig. 2.

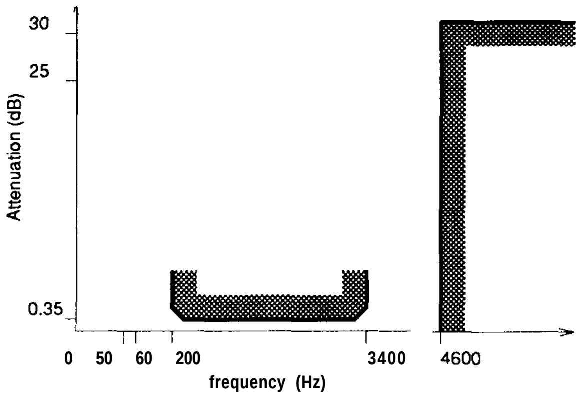


Fig.2. The CCITT receive specification.

The receive low-pass filter was to be used to compensate for the passband distortion caused by sinc roll-off. The desired response for the receive filter would no longer be a flat passband, but would now become an inverse-sine compensated passband satisfying a period equal to the sampling rate of the system. The overall response would then meet the desired specifications.

The specifications for both filters are challenging and pose a number of interesting difficulties for the designer. filtorX provides a versatile and systematic design solution.

2 filtorX

filtorX is an interactive language intended to design, manipulate, and synthesise filters. It is a high-level language based on C[2] and C++[l] which uses complex numbers, and rational functions as its basic data types. filtorX is contained within the fX environment which includes an X windows[6] interface consisting of menus and dialogue boxes, and a real-time plotting program. filtorX can be controlled either through the graphical interface or by entering commands from the keyboard directly to the language interface.

3 Transmit filter

Partitioning the specification: Since the transmit filter was a bandpass requiring a sharp cut-off, it was determined that separating the filter into lowpass and highpass sections would be necessary to obtain acceptable numerical accuracy. The specification can be easily separated by assigning the upper-stopband to a lowpass specification with the passband extending to DC. Similarly, the highpass filter specification would be created by assigning the lower-stopband of the bandpass specification and extending the passband to infinity. The separated specifications are displayed in Figure 3. The passband ripple can be divided evenly between the lowpass and highpass filters so that the cascade does not exceed tolerable limits.*

Applying filtorX to the lowpass filter: Having partitioned the transmit specification

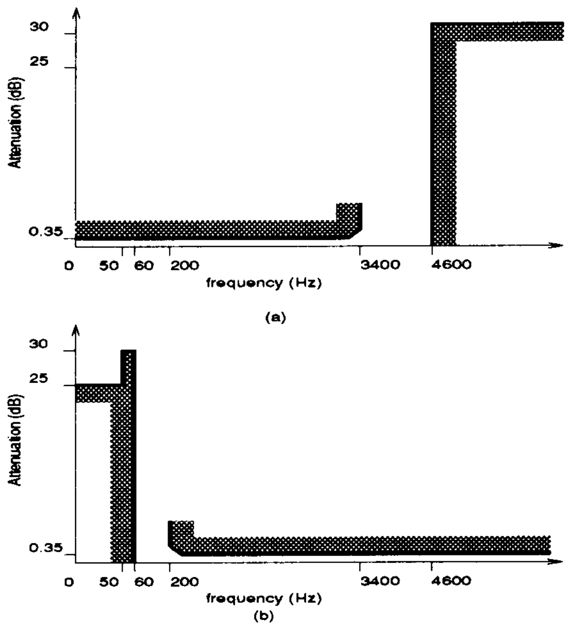


Fig.3. The partitions of the transmit specification: a) lowpass; h) highpass.

into lowpass and highpass portions, we can obtain transfer functions using the available Remez-exchange algorithm (applicable to either lowpass or bandpass filters). The lowpass portion of the specification can be entered via the 'editLowPass' menu selection from the 'specifications' menu of fX shown in Figure 4.

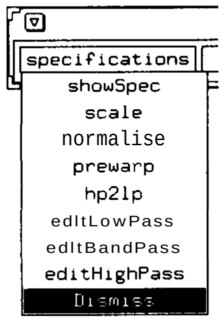
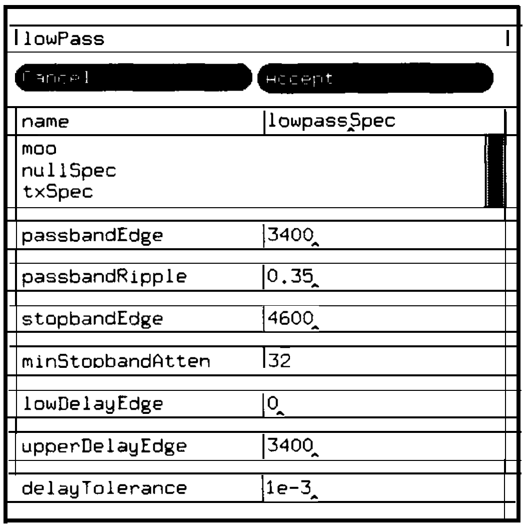


Fig.4. The 'specifications' main menu.

Creating a specification: The break-

* Although both the lowpass and highpass filters were assigned half of the passband ripple specified in this example, it may be advantageous to partition the ripple in order to reduce requirements of one of the filters and, possibly, reduce the order.

points from Figure 3a) are entered into the ‘lowPass’ dialogue box, as shown in Figure 5, to create a spec object in filtorX. Select-



name	lowpassSpec
moo	
nullSpec	
txSpec	
passbandEdge	3400
passbandRipple	0.35
stopbandEdge	4600
minStopbandAtten	32
lowDelayEdge	0
upperDelayEdge	3400
delayTolerance	1e-3

Fig.5 The ‘lowPass’ dialogue box.

ing the ‘showSpec’ option displays the low-pass specification as it would be defined in the filtorX syntax.

```
spec 1owpassSpec (passband (0,
3400, 0.175) stopband (4600,
Infinity, 32))
spec 1owpassSpec ( gdelayc  0,
3400,  le-3 1)
```

Normalising a specification: Before moving to the remez optimisation, it is important to normalise the specification for good numerical accuracy. This is done using the ‘normalise option of the ‘specifications’ menu to create a normalised lowpass specification.

Remez optimisation: The ‘Remez_Optimisation’ entry of the ‘rationalFunctions’ menu of Figure 6 yields a dialogue box as shown below in Figure 7. The remez algorithm can generate either equiripple or maxi-

The dialogue box includes group-delay information which is not applicable in this example, but has been altered from the default values to correspond to the given magnitude specifications.

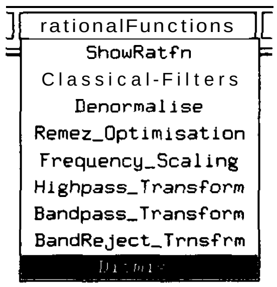
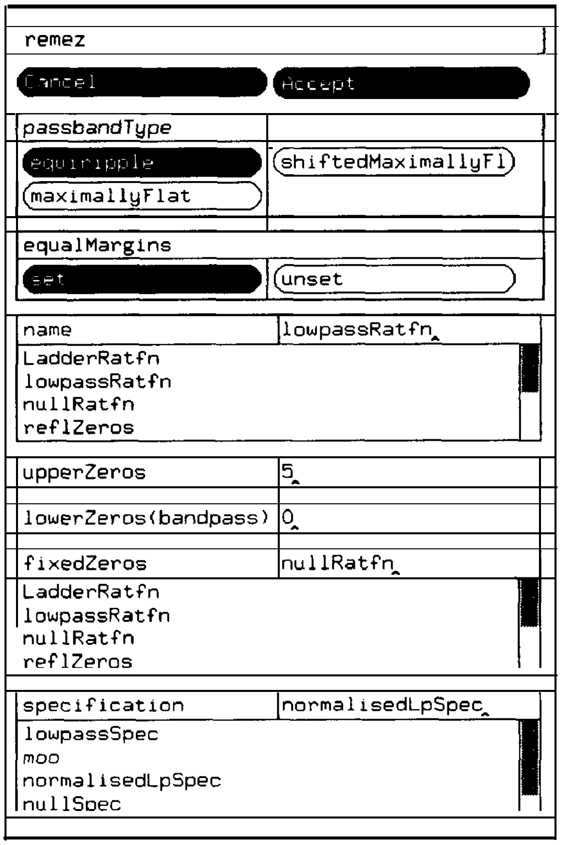


Fig.6. The ‘rationalFunctions’ main menu.



passbandType	
[✓ equiripple]	shiftedMaximallyFl
maximallyFlat	
equalMargins	
[✓ set]	unset
name	lowpassRatfn
LadderRatfn	
lowpassRatfn	
nullRatfn	
reflZeros	
upperZeros	5
lowerZeros(bandpass)	0
fixedZeros	nullRatfn
LadderRatfn	
lowpassRatfn	
nullRatfn	
reflZeros	
specification	normalisedLpSpec
lowpassSpec	
moo	
normalisedLpSpec	
nullSpec	

Fig.7. The ‘Remez_Optimisation’ dialogue box.

mally flat passbands (or a shifted version of a maximally flat passband for lowpass filters) along with an equiripple stopband; the equiripple choice is selected for a minimum order solution. A name is given to the resulting rational function and the number of zeros in the upper stopband (upperZeros) is entered (the number of zeros in the lower stopband, lowerZeros, is applicable to bandpass filters only). Finally, the applicable specification is selected.

Applying filtorX to the highpass filter: The highpass partition of the original specification is slightly more complex and requires that the specification be entered manually. filtorX can be operated graphically via menus and dialogue boxes or textually through the keyboard. Generally straight-forward functions can be easily controlled graphically whereas complex functions require designer control. Following the format shown earlier for the '1owpassSpec' specification, the highpass specification would be entered as follows,

```
spec  highpassSpec(
stopband(0,50,25)(50,60,30);pa
ssband(200,Infinity,0.175))
spec  highpassSpec(gdelay(
200,Infinity, 1E-3))
```

where the bold strings denote filtorX keywords.

Creating a lowpass prototype: Since the remez algorithm does not operate on highpass specifications, it is first necessary to create a lowpass prototype filter. The first step is to transform the highpass specification to an equivalent lowpass specification using the 'hp2lp' entry of the 'specifications' menu. The remaining steps are identical to those followed above to obtain the lowpass filter, 1owpassRatfn. It then remains to transform the lowpass prototype filter into a highpass filter using the 'Highpass-Transform' entry of the 'rationalFunctions' menu.

Cascading transfer functions: The desired bandpass filter is then obtained by cascading the lowpass and highpass filters. The final transfer function is determined by simply multiplying the two rational functions as,

```
bandpass = lowpass * highpass
```

which is analogous to cascading the two filter sections. The resulting transmit filter is plotted in Figure 8.

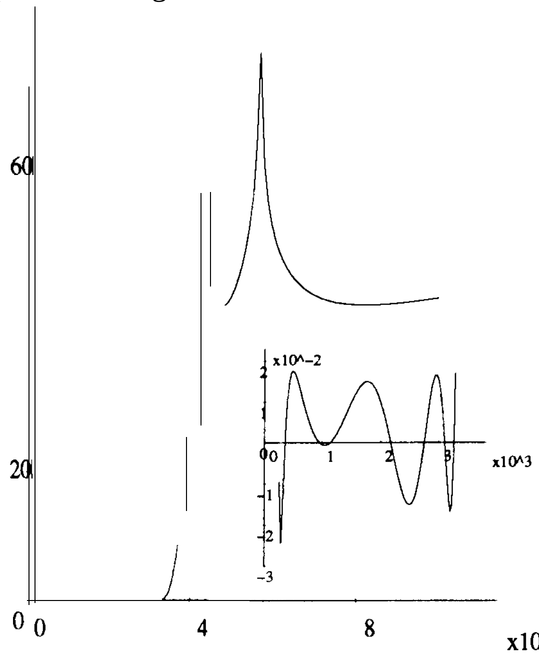


Fig.8 Attenuation response, in dB, versus frequency, in Hz, of the transmit filter. Passband focus in inset.

4 Receive filter

Transfer function optimisation: A classical elliptic rational function was used as the seed for an optimisation to obtain the inverse-sine compensated passband. The optimisation requires a user defined function as the desired response from which error values can be determined. In the passband, inverse-sine compensation is desired so we create a function in filtorX as

```
function invSinc(){
 if ($1==0){
  return 1
 } else {
  return ($1*PI/fs)/sin($1*PI/
fs)
 }
}
```

Note that the function returns a linear gain value, not a deciBe1 value. The ideal for the stopband is zero transmission necessitating the simple function,

```
function zero () return 0 .O
```

filtorX includes a least-pth optimiser[7], which is a generalisation of the least-mean squares algorithm with the option of specifying weights to control error tolerances. The receive filter obtained using the optimiser and the defined functions is plotted in Figure 9.

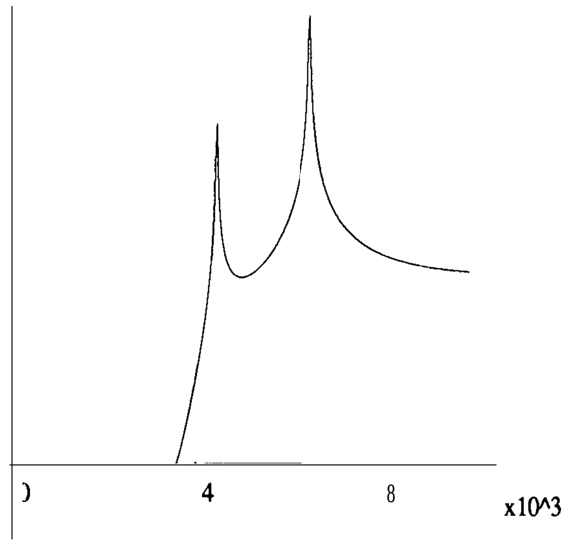


Fig.9. Attenuation response, in dB, versus frequency, in Hz, of the inverse-sine compensated receive filter.

5 Obtaining biquadratic sections

Given the transmit and receive transfer functions, it was determined that biquadratic sections would be used to implement the complete filter. This will facilitate the use of multiple sampling rates for the transmit filter; the reasons for this decision will be discussed in a following section. To optimise filter performance, it is necessary to ensure good noise characteristics and dynamic range for the overall structure. This is achieved through judicious choice of pairing of the poles and zeros of the original filters into biquadratic sections, then by appropriately ordering and scaling the biquadratic sections.

Figure 10 features an example of pole/

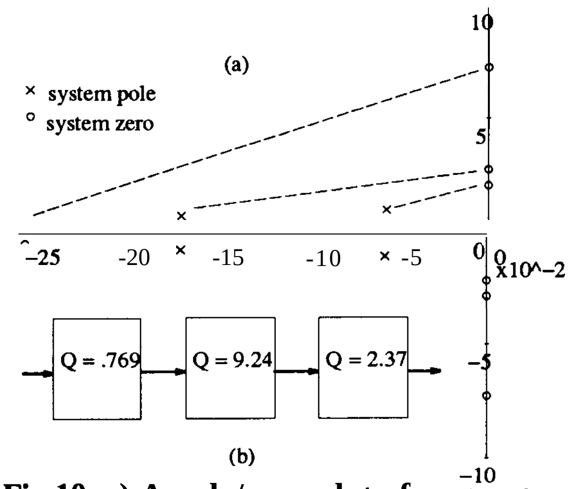


Fig.10. a) A pole/zero plot of a normalised 6th-order equiripple filter. b) The order and Q-values of the biquadratic sections ordered according to[9].

zero pairing and biquad ordering for a sixth-order lowpass filter.

Scaling for dynamic range: Once the biquad sections have been ordered, their individual gains can be set for maximal dynamic range. This consists of setting the peak gain of each biquad so that the composite gain at that intermediate stage of the circuit is always at the overall desired gain of the complete filter[8]. The individual biquads, as well as the composite results, are plotted in Figure 11 to illustrate the concepts.

Originally, the tasks of pairing, ordering, and scaling were performed through a combination of programs written in the Pascal programming language and by hand analysis following standard rules of thumb[8]. This entailed converting data and information from the format of one program to another, printing data out to be manipulated by the designer then re-entering new data into the computer. This is a tedious and time consuming task to be performed once, but in the pro-

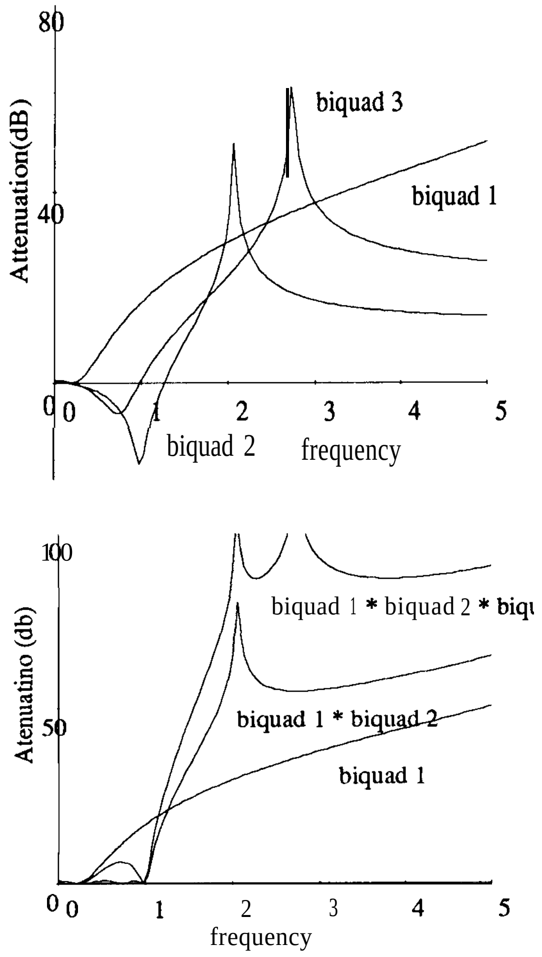


Fig.11. a) The magnitude responses of the individual biquads of Figure10 scaled for maximum dynamic range. b) The magnitude responses as each biquad is added to the cascade.

cess of design, it can be repeated many times before satisfactory results are achieved

Using filtorX, a procedure was written to pair the poles and zeros of a higher order function, creating biquadratic transfer functions (with one linear transfer function for odd ordered prototypes[4]). Capturing this knowledge in filtorX eliminates the interchange of data between various programs, eliminates manual intervention and emancipates the designer from having to learn many programs and languages to perform one design.

6 Tailoring a Graphical Interface

With the addition of a graphical interface, the actions of pole/zero pairing, biquad ordering, and gain-scaling can be combined behind the control of a menu selection (and a dialogue box to allow user input to the process). The graphical interface is part of the fX environment and is user programmable using a simple text file. The simplicity of customisation allows any designer to tailor an environment to a specific project. The advantage is that other designers need not be aware of the details of implementation when they can control the functions graphically. The menu of Figure 12 was created using **13**

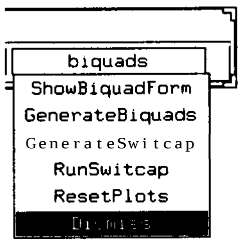


Fig.12. The biquadratic operations menu.

lines of code while two associated dialogue boxes (of a simple layout) also required only 13 lines.

7 Language Issues

Among the advantages of a user programmable language is the ability of the designer to look into the makings of any of the given procedures. This allows a designer to 1. learn the syntax and coding techniques of the language 2. note the methodology and algorithms used in the given procedure and 3. alter the procedure (in part or whole) for her own purposes.

Because all procedures, both library and user defined, are accessible, any user has the entire functionality available to her. This gives an interested designer an opportunity to look within the operation and uncover the details of the 'black box' typically presented. A designer can confirm that procedures conform to manual techniques, learn different techniques, or unearth bugs.

Having had the opportunity to see the methodology of a procedure, the designer can then alter the algorithm to achieve new or different functionality. This provides the option of implementing new ideas quickly and efficiently: altering existing code minimises the designer's investment in learning new coding techniques and syntax. In an industrial environment designers can complete designs quickly employing existing techniques, later tuning the algorithm to optimise performance. Alternately, filtorX can be used to simply acquire design knowledge from the existing procedures; this is what makes filtorX an educational tool as well — the property of self-documentation.

8 Practical Considerations

Following standard telecommunications practice, a clocking rate of 128kHz was chosen for the receive filter; however, because the transmit filter is a bandpass with a pair of zeros near 6OHz, the capacitors required would dominate the circuit area due to the large ratio of sampling rate to the frequency of the zeros. Reducing the sampling rate yields a proportional area savings. Separating the transmit filter into lowpass and highpass filter sections[‡] allows the lowpass portion to be operated at the 128kHz sampling rate and to act as an anti-aliasing filter for the highpass filter section operating at a lower sampling rate. A system diagram illustrates the structure in Fig. 13.

The lowpass partition allows the highpass filter to be operated at a sampling rate of

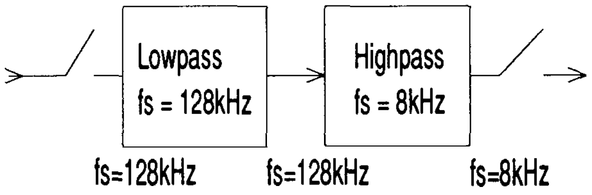


Fig.13. A block diagram of the multi-rate band-pass system. Sample-and-hold locations and sampling rates are noted.

8kHz, a decimation of sixteen times, which reduces capacitor sizes commensurately.

Originally, the designer performed the translation of the transfer function roots to various sampling rates manually, and simulated the results to determine the accuracy. Procedures in filtorX can perform the translation of the roots of the discrete-time transfer functions enabling experimentation with various decimation factors. The resulting cascade of the two discrete-time transfer functions can be determined by using a filtorX procedure[5].

The sampling rate of 8kHz chosen for the highpass section of the filter may result in some distortion of the response near 4kHz, the theoretical maximum of this sampled data system. The effects can be significant and can be responsible for the filter exceeding specified bounds.

Whereas WATSCAD[11] was originally used to simulate the circuits at their respective sampling rates, this can now be accomplished with the available procedures in filtorX. It is possible to note the discrepancy and correct for the error using optimisation in filtorX. The use of WATSCAD is still necessary to perform the design centering required for the realised production circuit.

Once suitable transfer functions are obtained, capacitor values for the actual circuits can be calculated. Since capacitor ratios can be obtained from the biquadratic transfer functions, it is a simple matter to create a filtorX procedure which does so for a given

‡ The separation is performed for a different purpose than that of section 2.

circuit topology; using this same information a switcap[10] netlist could also be generated. This was done for a given biquad[3] to illustrate the concept and could easily be extended to any other circuit topology.

9 Conclusions

A switched-capacitor filter design originally performed at Mite1 Semiconductor was studied to determine the methodology followed by the design team. It was noted that a significant amount of effort was devoted to writing programs (in Pascal) to perform complex design work, to translate data from one format to another, and to execute actions for which there was a known design methodology. filtorX was designed as a user programmable filter design language which allows designers to either write procedures in the high-level filtorX language, or use the provided procedures; both approaches were applied to the challenges of the project. In addition, the graphical capabilities were used to add a simple interface to the user defined actions. Since filtorX is intended for filter design, the language is suited to powerful expressions for filter manipulation; this was evidenced by examples of filtorX applied to various design problems. Its generality allows designers to perform most work within the filtorX environment. Coupled with simulators like WATSCAD or SWITCAP2, only the layout is left to complete the entire design.

Acknowledgments

This project is supported by ITRC, Micronet, and NSERC.

References

[1] M.A. Ellis, B. . *The Annotated C++ Reference Manual.* Addison Wesley Publishing Company, Reading, Massachusetts, 1990.

[2] B.W. Kernighan, D.M. Ritchie. *The C Programing Language.* Prentice-Hall inc., Englewood Cliffs, NJ, 1978.

[3] K. Martin, A.S. Sedra. Strays-insensitive switched-capacitor filters based on the bilinear z transform. *Electronics Letters.* vol. 19, pages 365-366, June, 1979.

[4] C. Ouslis, W.M. Snelgrove, A.S. Sedra. A Filter Designer's Filter Design Aid: filtorX. *Proceedings of the IEEE International Symposium On Circuits and Systems,* pages *376-379.* IEEE inc., Singapore, June 1991.

[5] C. Ouslis, W.M. Snelgrove, A.S. Sedra. Multi-rate switched-capacitor filter design with aggressive sampling-rates: filtorX in action. *Proceedings of the IEEE International Symposium On Circuits and Systems,* pages 1183-1186. IEEE inc., San Diego, May 1992.

[6] R.W. Scheifler, J. Getys. The X Window System. *ACM Transactions on Graphics. No. 63, 1986.*

[7] R. Schreier. *Transfer Function Design.* Master's thesis, University of Toronto, Department of Electrical Engineering, Toronto, Canada, 1985.

[8] A.S. Sedra, P.O. Brackett. *Filter theory and Design: Active and Passive.* Matrix Publishers inc., Beaverton, Oregon, 1978.

[9] W.M. Snelgrove, A.S. Sedra. Optimization of Dynamic Range in Cascade Active Filters. *Proceedings of the IEEE International Symposium On Circuits and Systems,* pages 151-155. 'IEEE inc., New York, May 1978.

[10] User's Guide for SWITCAP, Version 5., Columbia University, August, 1987.

[11] WATSCAD User's Guide, University of Waterloo

[12] J. Welsh, J. Elder. *Introduction to Pascal - Second Edition.* Prentice-Hall International, Englewood Cliffs, New Jersey, 1982.