

# Algorithm Analysis and Mapping Environment for Adaptive Computing Systems

Eric Pauer, Cory Myers, Ken Smith, and Paul Fiore  
 (pauer,cory,jmsmith,pfiore)@sanders.com

**Sanders, a Lockheed Martin Company**

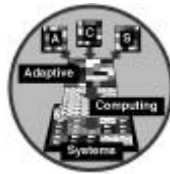
Nashua, NH 03061



Third Bi-Annual Ptolemy Miniconference - 1999

## Statement of the Problem

Reconfigurable computing technology offers leap ahead performance, e.g. 10X ops per watt and/or ops per cubic inch, over general purpose programmable solutions without the need to develop custom hardware. However, today generation of a working implementation requires hardware design expertise and generation of a good implementation requires many slow iterations between an algorithm designer and a hardware developer.



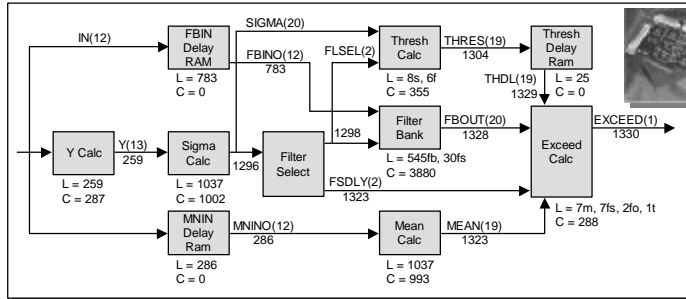
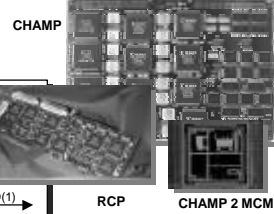
# Adaptive Computing Performance Gain

	CHAMP	TMS 320C80
Image Size	256 x 256	256 x 256
Implementation Time	44 Days	28 Days
Frame Rate	305 frames/sec	12 frames/sec
Latency	68 $\mu$ sec	82,000 $\mu$ sec
Processing Load	4.7 Bops	0.2 Bops
Utilization	73%	Unknown
Gates	510k	N/A

(Operation count increase by 70% if memory loads and stores are counted)

**Greater than 10X performance**  
Design time measured in weeks

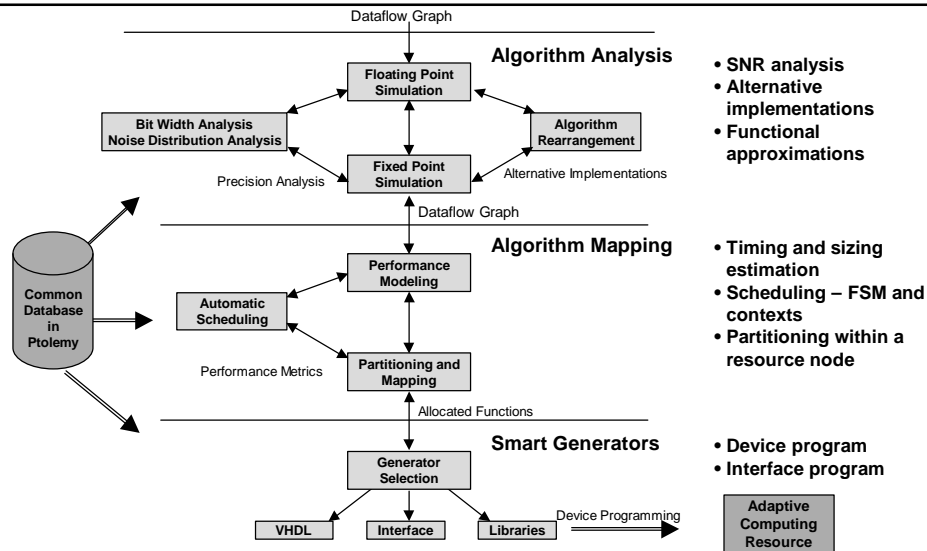
## Reconfigurable Architectures



**Analysis:**  
Bit Widths  
Latency (L)  
Cells Used (C)



# Analysis and Mapping in ACS Environment



- Algorithm Analysis**
- SNR analysis
  - Alternative implementations
  - Functional approximations

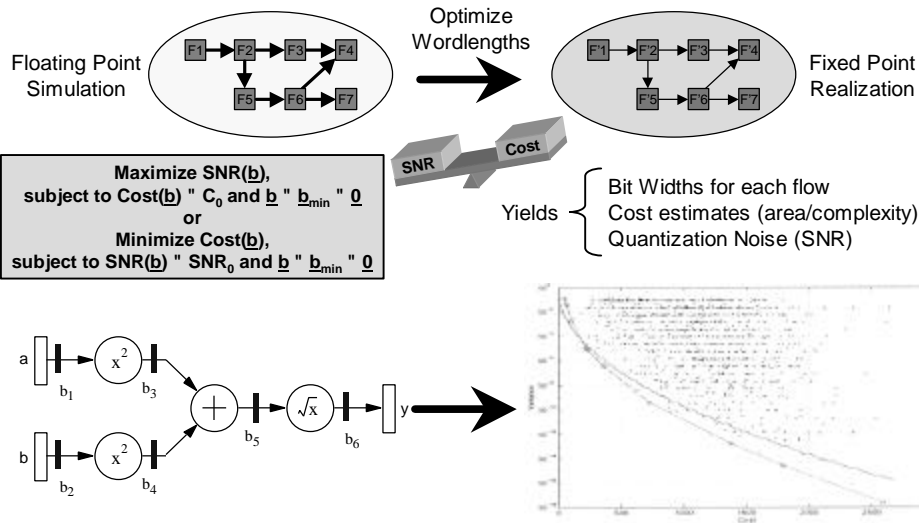
- Algorithm Mapping**
- Timing and sizing estimation
  - Scheduling – FSM and contexts
  - Partitioning within a resource node

- Smart Generators**
- Device program
  - Interface program

Adaptive Computing Resource

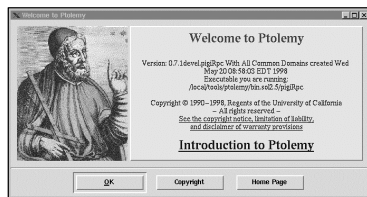


# Automated Float to Fixed Point Translation

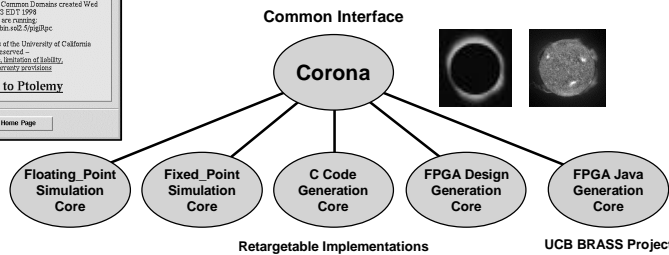


# Ptolemy and the ACS Domain

- Ptolemy - simulation/design environment from the University of California, Berkeley (<http://ptolemy.eecs.berkeley.edu>)
- New ACS domain developed to facilitate movement among simulation and code/design generation (released in 0.7.1, 6/98)
- ACS Stars (basic building block) are composed of a Corona (interface) and multiple cores (implementations)
- Core (implementation) selection is via targeting mechanism



Available cores/targets



# Top Level Example

- FIR filter to be implemented in both floating point and fixed point simulation

The screenshot displays a Ptolemy II schematic for a "Retargetable 16-tap FIR Filter". The schematic shows a sequence of 16 delay elements (represented by circles with 'z^-1') followed by a series of multipliers and an adder. A configuration dialog box is open, showing parameters for the filter:

- procid: -1
- OverflowHandler: validate
- ReportOverflow: NO
- RoundFix: YES
- OutputPrecision: prec
- ArrivingPrecision: YES
- InputPrecision: prec

Other dialog boxes provide detailed information about overflow handling and precision settings.



# Selecting Among Alternative Implementations

- Alternative implementations are represented as "targets"
- Targets can have parameters
- Floating point simulation, fixed point simulation, and C code generation are integrated today. FPGA generation almost ready.

The image shows two screenshots related to target selection. The left screenshot is a "Choose one:" dialog with three radio buttons: "default-ACS", "ACS-Fix", and "ACS-CGC". The "default-ACS" option is selected. Below the dialog, the text "ACS-CGFPGA soon!" is displayed. The right screenshot shows a C code generation window with the following code:

```

// User:         user
// Date:         Wed May 20 10:21:48 1998
// Target:       KCC32
// Universe:     butterfly */

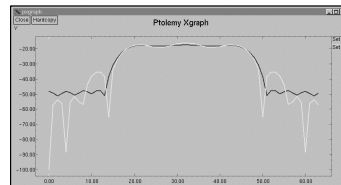
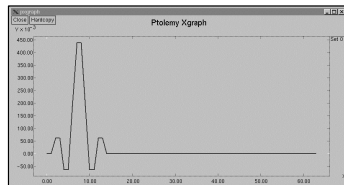
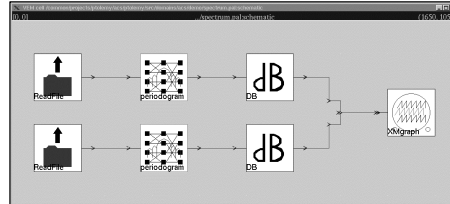
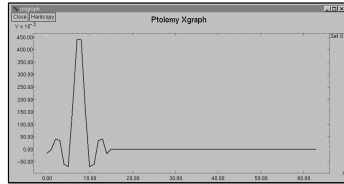
/* Define macros for prototyping functions on ANSI C non-WGI compilers */
#define ARG0
#define ARG1
#define ARG2
#define ARG3
#define ARG4
#define ARG5
#define ARG6
#define ARG7
#define ARG8
#define ARG9
#define ARG10
#define ARG11
#define ARG12
#define ARG13
#define ARG14
#define ARG15
#define ARG16
#define ARG17
#define ARG18
#define ARG19
#define ARG20
#define ARG21
#define ARG22
#define ARG23
#define ARG24
#define ARG25
#define ARG26
#define ARG27
#define ARG28
#define ARG29
#define ARG30
#define ARG31
#define ARG32
#define ARG33
#define ARG34
#define ARG35
#define ARG36
#define ARG37
#define ARG38
#define ARG39
#define ARG40
#define ARG41
#define ARG42
#define ARG43
#define ARG44
#define ARG45
#define ARG46
#define ARG47
#define ARG48
#define ARG49
#define ARG50
#define ARG51
#define ARG52
#define ARG53
#define ARG54
#define ARG55
#define ARG56
#define ARG57
#define ARG58
#define ARG59
#define ARG60
#define ARG61
#define ARG62
#define ARG63
#define ARG64
#define ARG65
#define ARG66
#define ARG67
#define ARG68
#define ARG69
#define ARG70
#define ARG71
#define ARG72
#define ARG73
#define ARG74
#define ARG75
#define ARG76
#define ARG77
#define ARG78
#define ARG79
#define ARG80
#define ARG81
#define ARG82
#define ARG83
#define ARG84
#define ARG85
#define ARG86
#define ARG87
#define ARG88
#define ARG89
#define ARG90
#define ARG91
#define ARG92
#define ARG93
#define ARG94
#define ARG95
#define ARG96
#define ARG97
#define ARG98
#define ARG99
#define ARG100
#define ARG101
#define ARG102
#define ARG103
#define ARG104
#define ARG105
#define ARG106
#define ARG107
#define ARG108
#define ARG109
#define ARG110
#define ARG111
#define ARG112
#define ARG113
#define ARG114
#define ARG115
#define ARG116
#define ARG117
#define ARG118
#define ARG119
#define ARG120
#define ARG121
#define ARG122
#define ARG123
#define ARG124
#define ARG125
#define ARG126
#define ARG127
#define ARG128
#define ARG129
#define ARG130
#define ARG131
#define ARG132
#define ARG133
#define ARG134
#define ARG135
#define ARG136
#define ARG137
#define ARG138
#define ARG139
#define ARG140
#define ARG141
#define ARG142
#define ARG143
#define ARG144
#define ARG145
#define ARG146
#define ARG147
#define ARG148
#define ARG149
#define ARG150
#define ARG151
#define ARG152
#define ARG153
#define ARG154
#define ARG155
#define ARG156
#define ARG157
#define ARG158
#define ARG159
#define ARG160
#define ARG161
#define ARG162
#define ARG163
#define ARG164
#define ARG165
#define ARG166
#define ARG167
#define ARG168
#define ARG169
#define ARG170
#define ARG171
#define ARG172
#define ARG173
#define ARG174
#define ARG175
#define ARG176
#define ARG177
#define ARG178
#define ARG179
#define ARG180
#define ARG181
#define ARG182
#define ARG183
#define ARG184
#define ARG185
#define ARG186
#define ARG187
#define ARG188
#define ARG189
#define ARG190
#define ARG191
#define ARG192
#define ARG193
#define ARG194
#define ARG195
#define ARG196
#define ARG197
#define ARG198
#define ARG199
#define ARG200
#define ARG201
#define ARG202
#define ARG203
#define ARG204
#define ARG205
#define ARG206
#define ARG207
#define ARG208
#define ARG209
#define ARG210
#define ARG211
#define ARG212
#define ARG213
#define ARG214
#define ARG215
#define ARG216
#define ARG217
#define ARG218
#define ARG219
#define ARG220
#define ARG221
#define ARG222
#define ARG223
#define ARG224
#define ARG225
#define ARG226
#define ARG227
#define ARG228
#define ARG229
#define ARG230
#define ARG231
#define ARG232
#define ARG233
#define ARG234
#define ARG235
#define ARG236
#define ARG237
#define ARG238
#define ARG239
#define ARG240
#define ARG241
#define ARG242
#define ARG243
#define ARG244
#define ARG245
#define ARG246
#define ARG247
#define ARG248
#define ARG249
#define ARG250

```

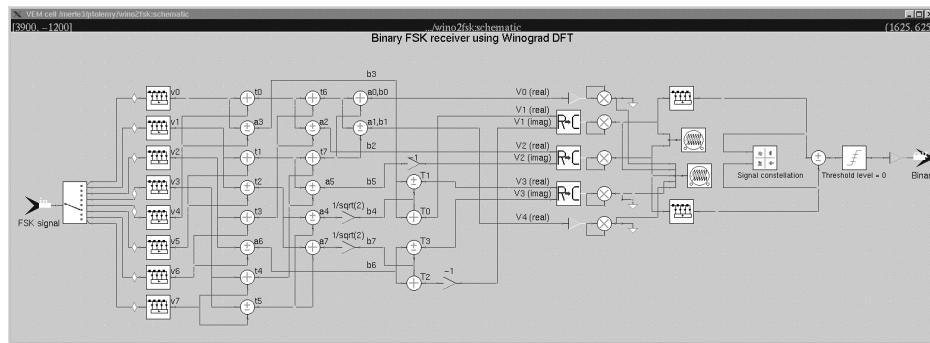
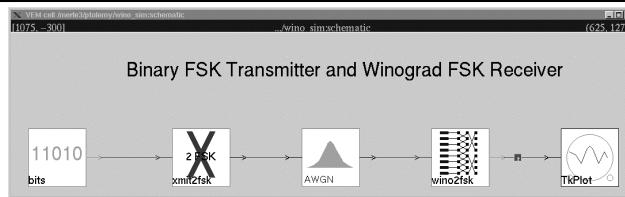


# Comparing Implementations

- Comparison of floating point and fixed point implementations

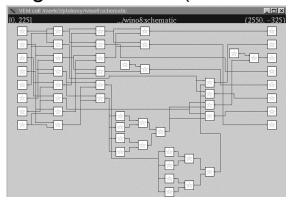


# Winograd-based FSK Receiver



# ACS Domain - CGFPGA Target

Winograd dataflow (ACS domain)



CGFPGA target yields:  
VHDL design  
and schedule

VHDL design (generated)

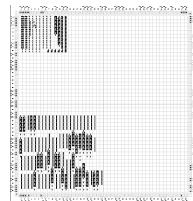
```

--completion: finish
Done <= '0';
Wait_High <= '1';
case Addr_State is
when Init_State =>
if (Counter = 1) then
Next_Addr_State <= Code01;
else
Next_Addr_State <= Init_State;
end if;
when Code01 =>
Next_Addr_State <= Code02;
MC_Addr_Control <= '1';
when Code02 =>
Next_Addr_State <= Code03;
--address generator (adder) preload generator
ADDR CLR <= '1';
when Code03 is
if (MC_Carry(8)='0') then
Next_Addr_State <= Init_State;
Done <= '1';
Wait_High <= '0';
else
--word counter control generator
MC_ADDR <= '1';
--address count enable engaged
Addr_Addr_CE <= '1';
Next_Addr_State <= Code03;
end if;
end case;
end adder;
    
```



Dataflow/Hardware schedule

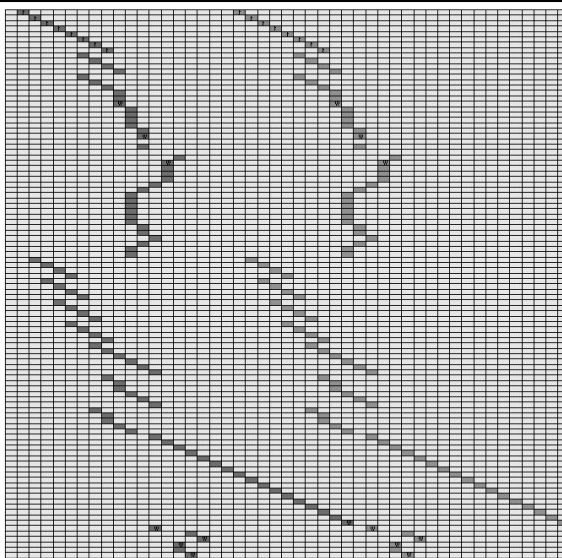
The results are sent to synthesis  
and place/route, yielding  
complete FPGA implementation!



# Winograd schedule

```

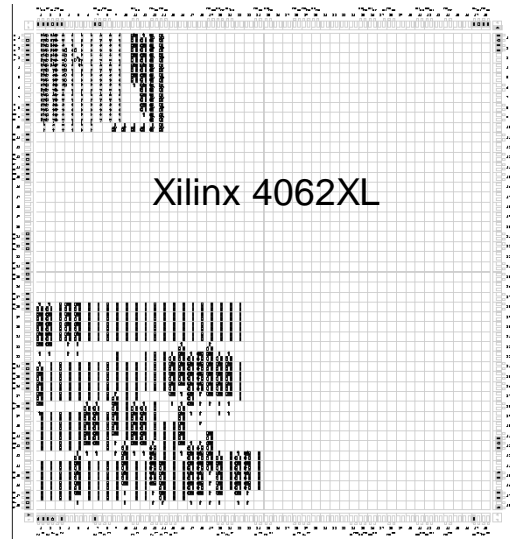
--completion: finish
Done <= '0';
Wait_High <= '1';
case Addr_State is
when Init_State =>
if (Counter = 1) then
Next_Addr_State <= Code01;
else
Next_Addr_State <= Init_State;
end if;
when Code01 =>
Next_Addr_State <= Code02;
MC_Addr_Control <= '1';
when Code02 =>
Next_Addr_State <= Code03;
--address generator (adder) preload generator
ADDR CLR <= '1';
when Code03 is
if (MC_Carry(8)='0') then
Next_Addr_State <= Init_State;
Done <= '1';
Wait_High <= '0';
else
--word counter control generator
MC_ADDR <= '1';
--address count enable engaged
Addr_Addr_CE <= '1';
Next_Addr_State <= Code03;
end if;
end case;
end adder;
    
```



# FPGA-based Winograd DFT

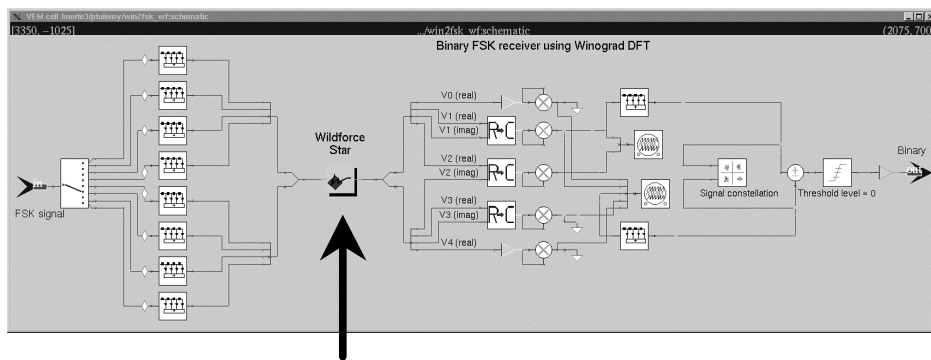
Address generator  
Word counter  
Data multiplexer  
Sequencer/State Machine

Winograd DFT  
computations



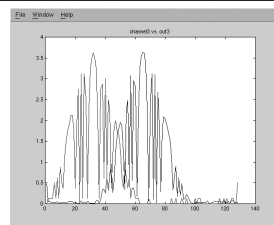
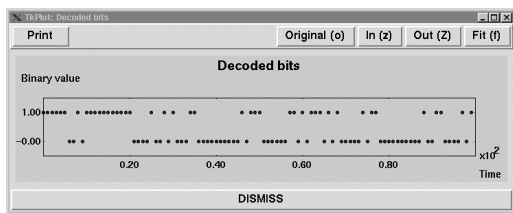
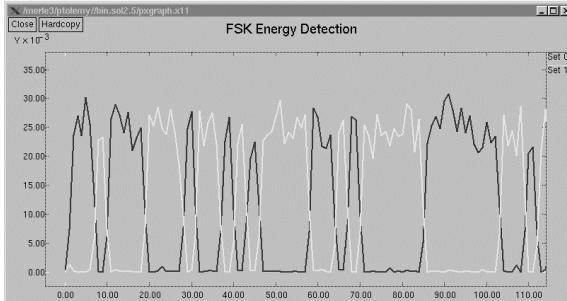
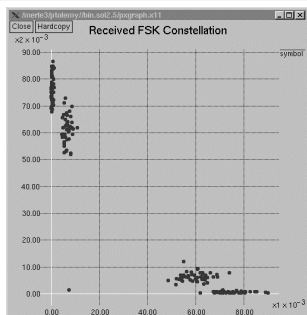
# Hardware-in-the-loop

## SDF Galaxy



SDF Wildforce star executes complete FPGA design  
in hardware on Annapolis Wildforce FPGA board





Sim vs. Hardware



## Related ACS Work at Sanders

