

01000111 01000101 01000100 01000001 01000101
Gedae, Inc.
01000111 01000101 01000100 01000001 01000101
01000111 01000101 01000100 01000001 01000101
01000111 01000101 01000100 01000001 01000101

Issues and Plans for Programming FPGAs and Alternate Architectures in Gedae

Jim Steed

March 6, 2003

Gedae, Inc.

jsteed@gedae.com

Goal

- Learn how developers currently use FPGAs and alternate architectures.
- Provide an integrated solution that automates that process.
- Problem: no broadly accepted development process

Issues

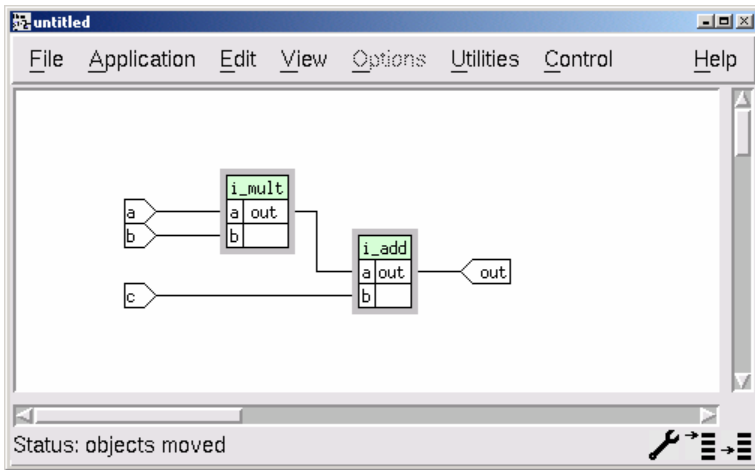
- Integration
 - Specification of Hardware in Gedae
 - Communication
 - Data Types
- Performance
 - Integration of Processing Cores
 - Expression of Parallelism

Gedaac, Inc. Issues (cont.)

- Performance (cont.)
 - Mode Control and Segmented Data
 - Easier Development of Larger Devices
 - Handling Alternate Implementations
- Analysis
 - Trace Events
 - True Simulation

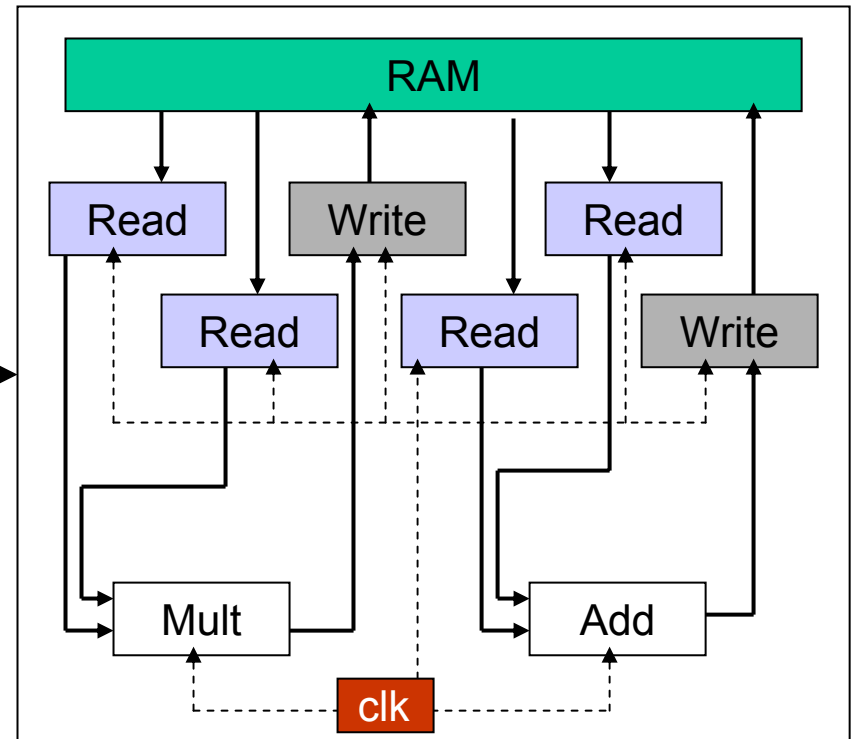
Specification

- How do we make this translation?
- Is this a good translation to make?



Schedule

FPGA



Register Transfer Language

- Graphical hardware description language
- New primitive type: single sample
- Code specified in meta-language
- Four functions
 - Assignment `A(expression, variable)`
 - Decimate `D(variable, signal)`
 - Clock `C(input, signal)`
 - Register `R(input, output, signal)`

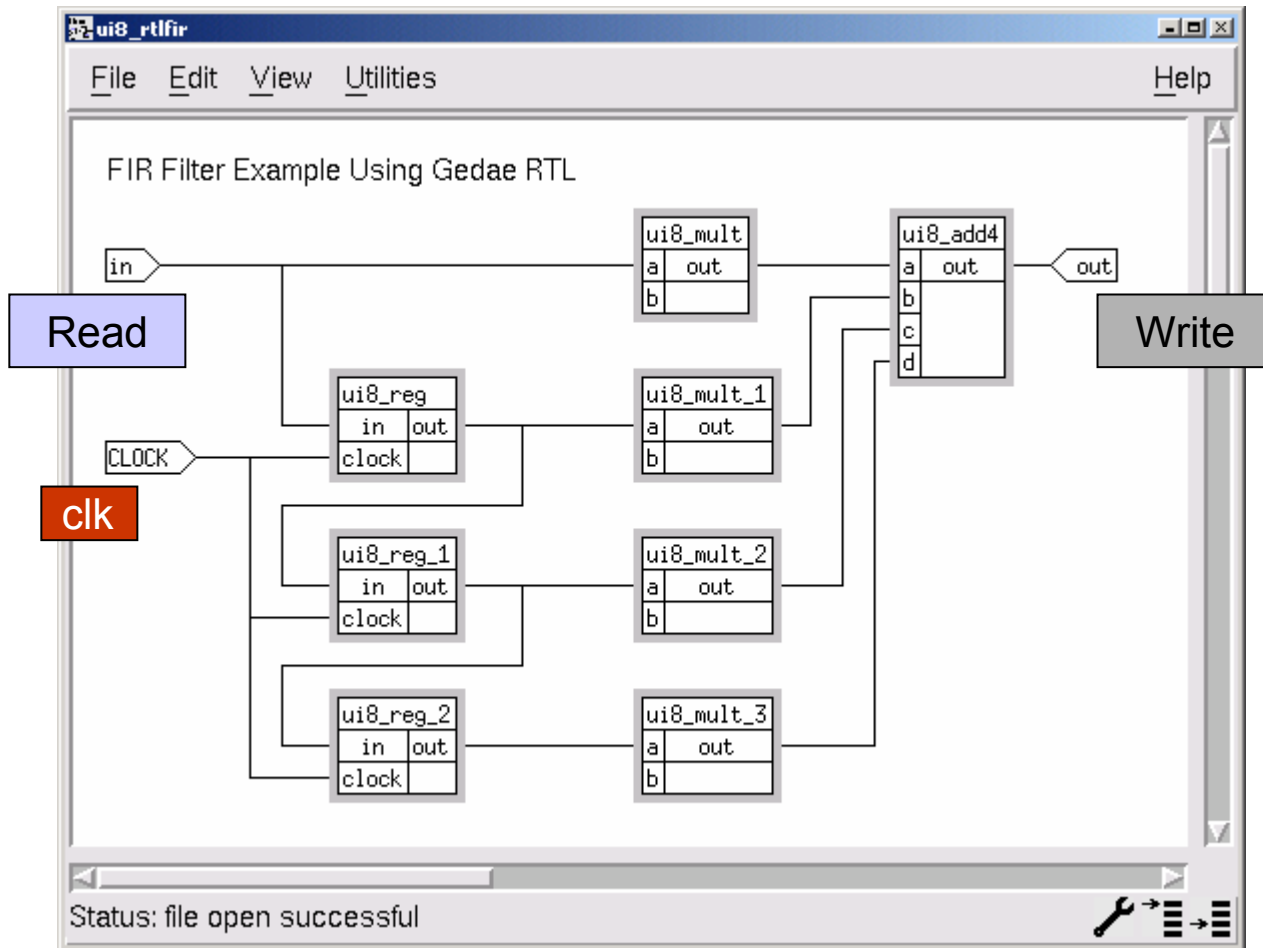
Special Data Types

- Fixed-width integers
- 1 to 32 bits
- Signed or unsigned
- When received on host, pad up to
 - char, unsigned char
 - short, unsigned short
 - int, unsigned int
- Use fixed-width ints to build other types

01000111 01000101 01000100 01000001 01000101
01000111 01000101 01000100 01000001 01000101
01000111 01000101 01000100 01000001 01000101
01000111 01000101 01000100 01000001 01000101

Gedae, Inc.

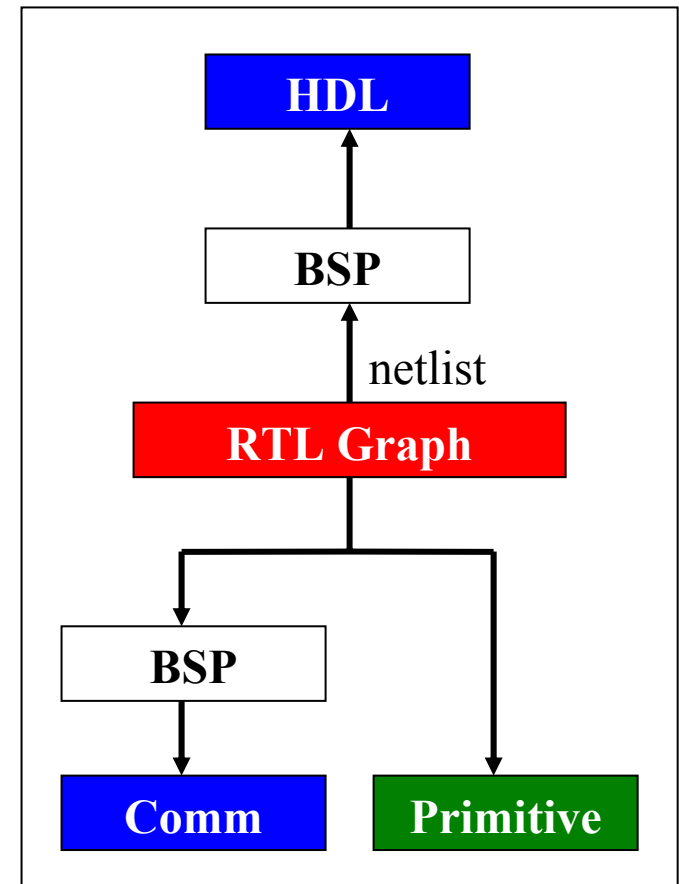
RTL Example



```
Name: ui8_mult
Type: single
sample
Input: {
    uint8 a;
    uint8 b;
}
Output: {
    uint8 out;
}
Function: {
    A(a*b, out);
}
```

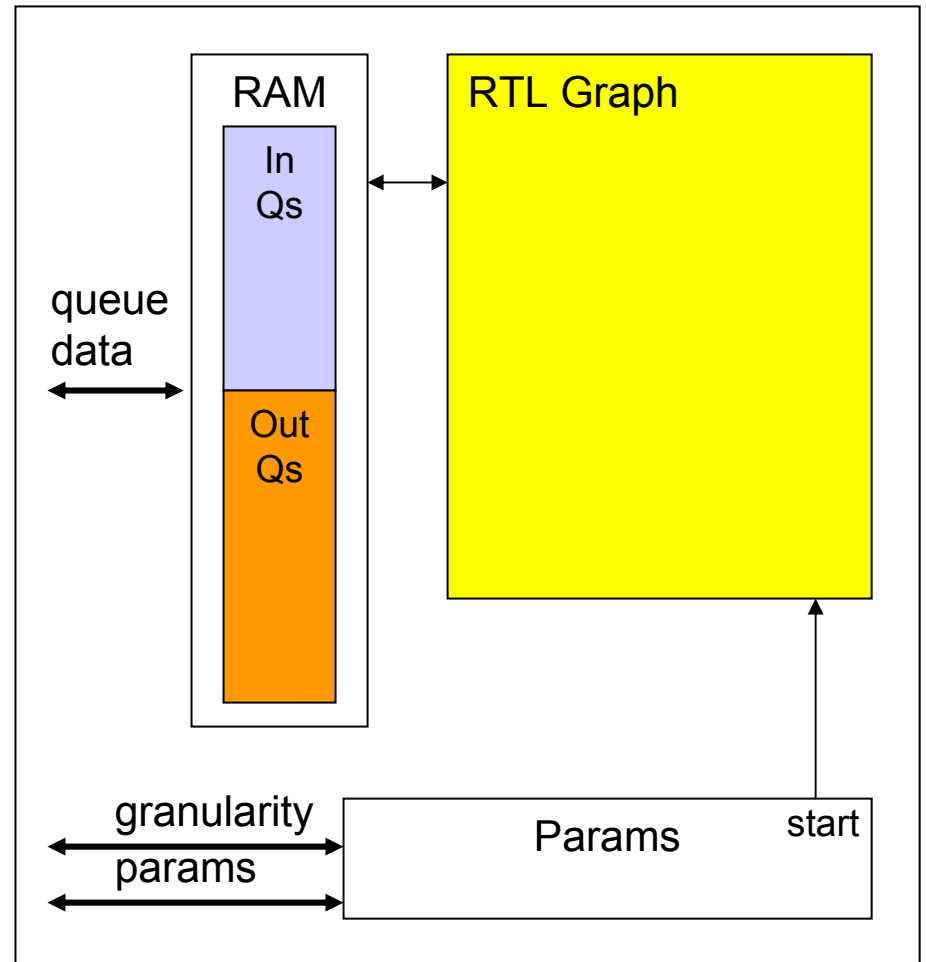
RTL Code Generation

- Hardware description languages via netlist
 - VHDL
 - System-C
 - Handel-C
- Stream primitive
 - Simulate algorithm, portable
 - Parent processor-side comm



Architecture

- One RTL graph forms one device
- Memory map created during compilation
- Simple protocol between parent proc and FPGA



Cores

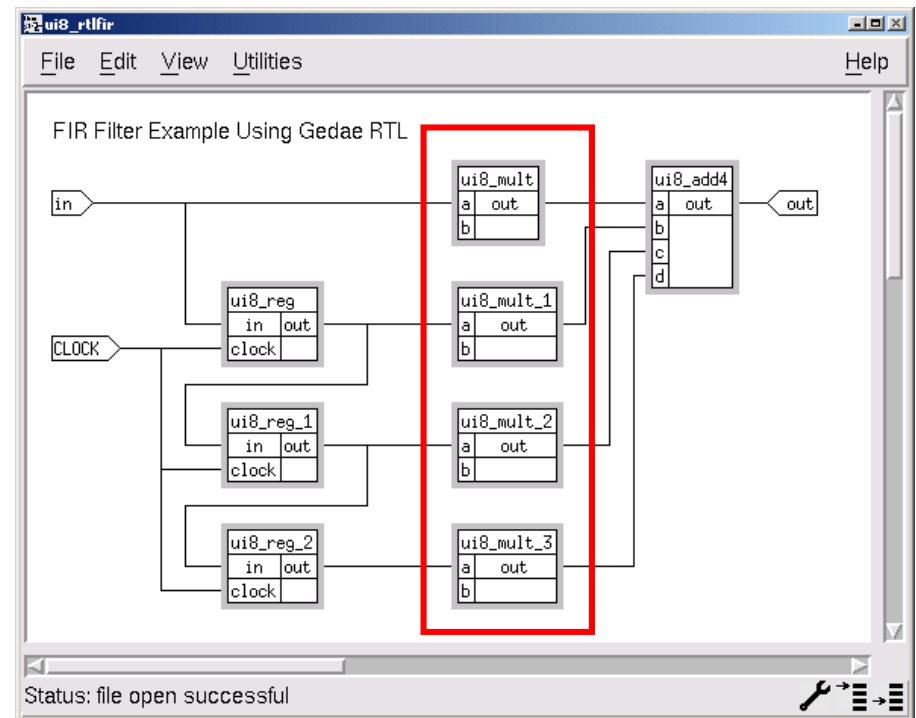
- Cores are efficient, pre-built implementations of common algorithms.
- Similar to use of optimized libraries (SAL, SKYvec, etc.)
- Provide solution similar to `e_library`:
replace subgraph with core if available,
otherwise export meta-language
- Must be expandable and customizable

Expression of Parallelism

- Exporting to C-based HDLs can impose serialism – must be specified directly
- The addition of a vector of length N can be implemented on an FPGA in three ways:
 - Use one adder to sequentially add the numbers, taking N clock cycles.
 - Use K adders, taking $\text{ceiling}(N/K)$ clock cycles.
 - Use N adders, taking one clock cycle.

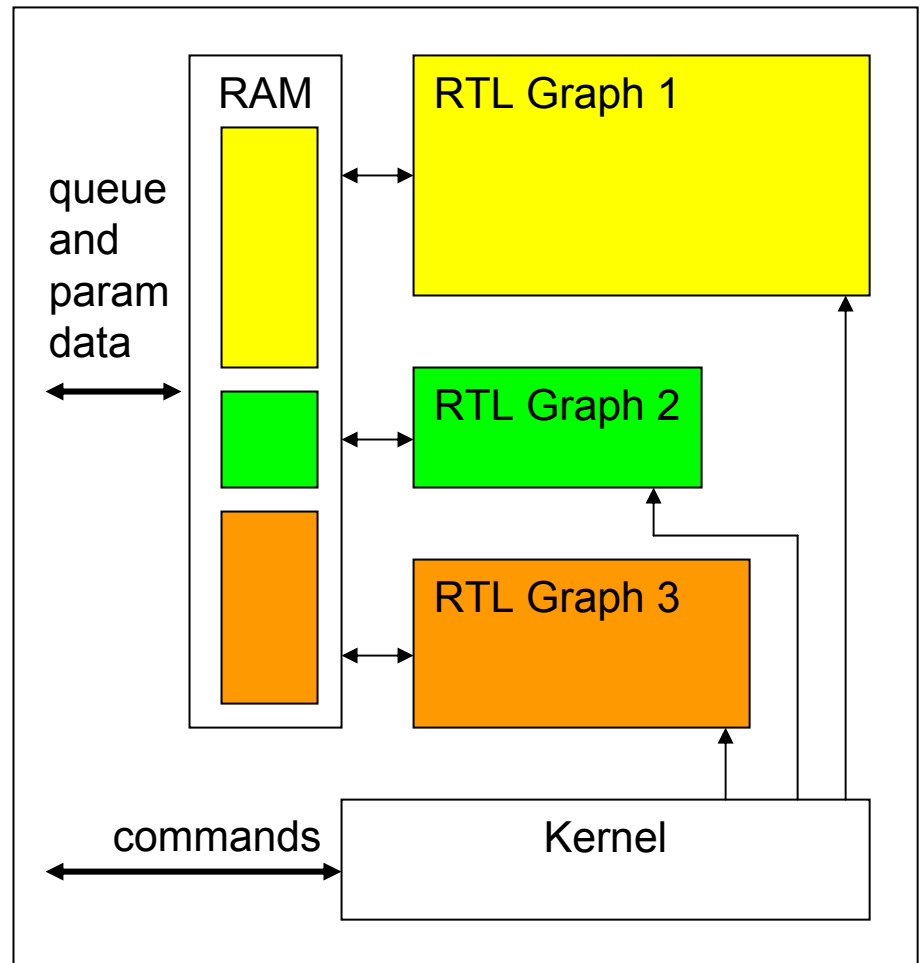
Analyze RTL Parallelism

- Analyze RTL to find maximal parallelization
- In FIR, ui8_mult are parallelizable
- Provide table for customization
- Export with netlist



Larger Devices

- Can RTL create large devices?
- Allow multiple RTL graphs on one FPGA
- Reduced-operation runtime kernel on the FPGA



Alternate Implementations

- Two implementations: RTL primitive and core library primitive
 - Port RTL to FPGAs and processors with vectorizing compilers
 - Port core library to AltiVec, SHARC, PowerPC, etc. processors
- Integrate switching between alternate implementations into Map Parts dialog

Analysis Tools

- Trace events
 - Collect in parent processor control
 - Provide 2nd FPGA kernel core with events
- GSIM
 - Bit-true simulation
 - Analyze area and depth
 - Use analysis data from vendor tools

Summary

- Register transfer language for specifying hardware in Gedae
- Export RTL to HDL and to stream primitive
- Performance: cores, parallelism, kernel
- Alternate implementations
- Analysis: trace events, GSIM
- Goal: Provide an integrated solution that automates the current development process