A Computing Model for Large-Scale Reconfigurable Systems

Carl Ebeling, Scott Hauck, Larry Ruzzo, Corey Olson, Maria Kim, Cooper Clausen, Boris Kogon
University of Washington
A Computing Model for Large-Scale Reconfigurable Systems

Or “How do you program 10,000 FPGAs?”
A Talk in Two Parts

- **Part I**
  - Genome Reassembly Problem

- **Part II**
  - A new programming model for reconfigurable computing
Re-sequencing a Genome

3 billion base pairs
Cut the Genome into Short Pieces

3 billion base pairs
“Read” the Short Sequences

3 billion base pairs

~ 1 week
Produces 200 million Short Reads

3 billion base pairs

~ 1 week

200 million short reads (76bp)
Reassemble Short Reads into Genome

~ 1 week

200 million short reads (76bp)

24-36 hours

3 billion base pairs
Reassembly Using a Reference Genome

- Use a reference genome as a scaffold
- Very small difference between individual genomes
  - SNPs
  - CNVs
  - indels
  - Total is < 1%
- Read error rate is ~1%
Reference

ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGGCTAGD

Short Reads

CGCTAGCGATATA
CATTACGCTAGCG
TATATACACGTGA
CGCGAACC GCCCTC
CGAGATCGCGCGT
GCGCGTACTACGCG
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGGTACTGTACGCGAACCGCCTCATATCGGCTAGDCGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACCGCCTC
CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACCGCCTC
CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGA
CGATATATACACGGA
TCGCTAGCGA
CTGTACGCG
AACCGCCTC
ATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATACTACGTGA
CGCGAACC
CGCTTC
CGAGATCGCG
CGT
CGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACCGAGCGATATATACACGCGATCGCGGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACC GCCCTC
CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACCGCCTC
CGAGATCGCGCGT
GCGCCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCAGATCGCGCTACTGTACGCGAACCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACCCTCTC
CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACCgcCCTC
CGAGATCGCGCGT
GCGGTACTACGC
TACACGAGATCGC
ATTCCACATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA

CATTACGCTAGCG
TATATACACGTGA
CGCGAACC CGCCTC
CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
 CGCTAGCGATATA CGCGAACCGCCTC
 CATTACGCTAGCG TATATACACGTGA

CGAGATCGCGCGT
GCGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA CGAGATCGGCGT CGCGAAACGCCTC
CATTACGCTAGCG TATATACACGTGA

GCWGCGTACTACGC
TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTACGATATA CGAGATCGCGCGT CGCGAACCCTTC
CATTACGCTAGCG TATATACACGTGA GCGGTAC--TACGC

TACACGAGATCGC
ATTCACCATTACGCGAGCGATATATACACGCGATCGCGCGTACTGTACGCGAACCGCCTCATATCGGCTAGD
CGCTAGCGATATA CGAGATCGCGCGT CGCGAACCCGCTC
CATTACGCTAGCG TATATACACGTGA GCGGTAC--TACGC
TACACGAGATCGC
Avoiding Brute Force Comparison

- Look up short reads in an index
  - A hash table of all subsequences of length 76

- Fails because of differences

- Solution: Use short subsequences to avoid differences
## Reference Index Table

- **Precompiled Hash Table**
  - Maps each short subsequence to locations in genome
  
- **Example for length 4 subsequences**

<table>
<thead>
<tr>
<th>Subsequence</th>
<th>Candidate Alignment Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
</tr>
<tr>
<td>AAAC</td>
<td>254</td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
</tr>
<tr>
<td>AAAT</td>
<td>47</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>AATC</td>
<td>48</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>TTTG</td>
<td>24</td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
</tr>
</tbody>
</table>
Using Seeds to Avoid Differences

Short Read
AAAATCGAATCGA

<table>
<thead>
<tr>
<th>Candidate Alignment Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
</tr>
<tr>
<td>AAAC</td>
</tr>
<tr>
<td>AAAG</td>
</tr>
<tr>
<td>AAAT</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>AATC</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>ATCG</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>TCGA</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>TTTG</td>
</tr>
<tr>
<td>TTTT</td>
</tr>
</tbody>
</table>
### Using Seeds to Avoid Differences

#### Seed

<table>
<thead>
<tr>
<th>Seed</th>
<th>Candidate Alignment Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
</tr>
<tr>
<td>AAAC</td>
<td>254 591</td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
</tr>
<tr>
<td>AAAT</td>
<td>47 174 822</td>
</tr>
<tr>
<td>...</td>
<td>... ... ... ... ... ... ...</td>
</tr>
<tr>
<td>AATC</td>
<td>48 823</td>
</tr>
<tr>
<td>...</td>
<td>... ... ... ... ... ... ...</td>
</tr>
<tr>
<td>ATCG</td>
<td>135 724</td>
</tr>
<tr>
<td>...</td>
<td>... ... ... ... ... ... ...</td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
</tr>
<tr>
<td>...</td>
<td>... ... ... ... ... ... ...</td>
</tr>
<tr>
<td>TTTG</td>
<td>24 201</td>
</tr>
<tr>
<td>TTTT</td>
<td>23 200 275 301</td>
</tr>
</tbody>
</table>
Using Seeds to Avoid Differences

Candidate Alignment Locations

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Location 1</th>
<th>Location 2</th>
<th>Location 3</th>
<th>Location 4</th>
<th>Location 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAC</td>
<td>254</td>
<td>591</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAT</td>
<td>47</td>
<td>174</td>
<td>822</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
<td>724</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTTG</td>
<td>24</td>
<td>201</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
<td>200</td>
<td>275</td>
<td>301</td>
<td></td>
</tr>
</tbody>
</table>
Using Seeds to Avoid Differences

<table>
<thead>
<tr>
<th>Candidate Alignment Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
</tr>
<tr>
<td>AAAC</td>
</tr>
<tr>
<td>AAAG</td>
</tr>
<tr>
<td>AAAT</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>AATC</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>ATCG</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>TCGA</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>TTTG</td>
</tr>
<tr>
<td>TTTT</td>
</tr>
</tbody>
</table>

AAAATCGAAATCGA
AAAA
AAAT
AATC
Using Seeds to Avoid Differences

Candidate Alignment Locations

<table>
<thead>
<tr>
<th></th>
<th>632</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAC</td>
<td>254</td>
<td>591</td>
<td></td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAT</td>
<td>47</td>
<td>174</td>
<td>822</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AATC</td>
<td>48</td>
<td>823</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
<td>724</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTTG</td>
<td>24</td>
<td>201</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
<td>200</td>
<td>275</td>
</tr>
</tbody>
</table>
Using Seeds to Avoid Differences

Candidate Alignment Locations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>632</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td></td>
<td>254</td>
<td>591</td>
<td></td>
</tr>
<tr>
<td>AAAC</td>
<td></td>
<td>361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAG</td>
<td></td>
<td>47</td>
<td>174</td>
<td>822</td>
</tr>
<tr>
<td>AAAT</td>
<td></td>
<td>48</td>
<td>823</td>
<td></td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
<td>724</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATCG</td>
<td></td>
<td>725</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCGA</td>
<td></td>
<td>24</td>
<td>201</td>
<td></td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
<td>200</td>
<td>275</td>
<td>301</td>
</tr>
</tbody>
</table>
Alignment of Each Short Read

<table>
<thead>
<tr>
<th>Candidate Alignment Locations</th>
<th>CAL</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
<td></td>
</tr>
<tr>
<td>AAAC</td>
<td>254</td>
<td>591</td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
<td></td>
</tr>
<tr>
<td>AAAAT</td>
<td>47</td>
<td>174</td>
</tr>
<tr>
<td>AATC</td>
<td>48</td>
<td>823</td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
<td>724</td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
<td></td>
</tr>
<tr>
<td>TTTG</td>
<td>24</td>
<td>201</td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
<td>200</td>
</tr>
</tbody>
</table>
Alignment of Each Short Read

Candidate Alignment Locations

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAC</td>
<td>254</td>
<td>591</td>
<td></td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAAT</td>
<td>47</td>
<td>174</td>
<td>822</td>
</tr>
<tr>
<td>AATC</td>
<td>48</td>
<td>823</td>
<td></td>
</tr>
<tr>
<td>ATCG</td>
<td>135</td>
<td>724</td>
<td></td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTTG</td>
<td>24</td>
<td>201</td>
<td></td>
</tr>
<tr>
<td>TTTT</td>
<td>23</td>
<td>200</td>
<td>275</td>
</tr>
</tbody>
</table>

Filter

CAL

Reference Lookup

Reference Subsequence
Alignment of Each Short Read

Candidate Alignment Locations

<table>
<thead>
<tr>
<th>Short Read</th>
<th>Reference Subsequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>632</td>
</tr>
<tr>
<td>AAAC</td>
<td>254 591</td>
</tr>
<tr>
<td>AAAG</td>
<td>361</td>
</tr>
<tr>
<td>AAAT</td>
<td>47 174 822</td>
</tr>
<tr>
<td>AATC</td>
<td>48 823</td>
</tr>
<tr>
<td>ATCG</td>
<td>135 724</td>
</tr>
<tr>
<td>TCGA</td>
<td>725</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>TTTG</td>
<td>24 201</td>
</tr>
<tr>
<td>TTTT</td>
<td>23 200 275 301</td>
</tr>
</tbody>
</table>
Mapping Alignment to Hardware

- Relatively straightforward algorithm
- Lots of data
- Lots of computation
- Lots of available parallelism
- Perfect for reconfigurable computing
Mapping Alignment to Hardware

- Relatively straightforward algorithm
- Lots of data
- Lots of computation
- Lots of available parallelism
- Perfect for reconfigurable computing

- So why does it have to be so hard?!
A Model for Programming Reconfigurable Computers

- What we are doing is closer parallel and distributed computation than traditional hardware design

- Why not leverage all the work that’s already been done?
Foundation: “Object” Model

- We call HW components “Objects”
  - More in common with SW objects than HW modules

- A system is comprised of Objects
  - Objects are static – constructed at compile-time
  - Constructors execute at compile-time

- Objects comprise:
  - Data/State that the object manages
  - Methods
Objects

- Object Data
  - Variables (registers)
  - Arrays (memory)
  - Object data is shared by the object methods
Objects

- **Object Data**
  - Variables (registers)
  - Arrays (memory)
  - Object data is shared by the object methods

- **Methods**
  - Methods are concurrently running HW threads
    - Methods all begin at system start
  - A method waits for a method call
    - Executes with provided parameters
    - Waits for next call
Method Calls

- Objects interact via method calls (ala RPC)
  - object.method(parameter-list)
  - Sends parameters from source method to destination method
  - Via a “connection”
    - Direct or time-multiplexed bus/network
Method Calls

- **Asynchronous methods – “Fire and Forget”**
  - a la Active Messages
  - No return – caller continues immediately
  - Return values forwarded via callback/callforward method
    - (explicit or implicit)

- **Synchronous methods**
  - Caller blocks until method completes (with return value)
  - Expensive except for local method calls
Simple Example

bar:
   foo.init();
   M.mult(a, b);
   M.mult(c, d);

M:
   mult(int x, int y) {
      foo.add(x * y)
   }

foo:
   int sum;
   SYNC init() {
      sum = 0;
   }
   add(int val) {
      sum += val;
   }
Simple Example

```plaintext
bar:
  foo.init();
  M.mult(a, b);
  M.mult(c, d);

M:
  mult(int x, int y) {
    foo.add(x * y)
  }

foo:
  int sum;
  SYNC init() {
    sum = 0;
  }
  add(int val) {
    sum += val;
  }
```
Simple Example

bar:
  foo.init();
  M.mult(a, b);
  M.mult(c, d);

M:
  mult(int x, int y) {
    foo.add(x * y)
  }

foo:
  int sum;
  SYNC init() {
    sum = 0;
  }
  add(int val) {
    sum += val;
  }
Simple Example

bar:
    foo.init();
    M.mult(a, b);
    M.mult(c, d);

M:
    mult(int x, int y) {
        foo.add(x * y)
    }

foo:
    int sum;
    SYNC init() {
        sum = 0;
    }
    add(int val) {
        sum += val;
    }
Simple Example

```java
bar:
    foo.init();
    M.mult(a, b);
    M.mult(c, d);

M:
    mult(int x, int y) {
        foo.add(x * y)
    }

foo:
    int sum;
    SYNC init() {
        sum = 0;
    }
    add(int val) {
        sum += val;
    }
```
Simple Example (cont)

M1:
    mult(int x, int y) {
        foo.add(x * y)
    }

bar:
    foo.init();
    M1.mult(a, b);
    M2.mult(c, d);

M2:
    mult(int x, int y) {
        foo.add(x * y)
    }

foo:
    int sum;
    SYNC init() {
        sum = 0;
    }
    add(int val) {
        sum += val;
    }
Simple Example (cont)

foo:
int sum;
SYNC init() {
    sum = 0;
}
add(int val) {
    sum += val;
}

M1:
mult(int x, int y) {
    foo.add(x * y)
}

bar:
    foo.init();
    M1.mult(a, b);
    M2.mult(c, d);

M2:
mult(int x, int y) {
    foo.add(x * y)
}
M1:

```java
mult(int x, int y) {
    foo.add(x * y)
}
```

bar:

```java
foo.init();
M1.mult(a, b);
M2.mult(c, d);
```

M2:

```java
mult(int x, int y) {
    foo.add(x * y)
}
```

foo:

```java
int sum;
SYNC init() {
    sum = 0;
}
add(int val) {
    sum += val;
}
```
Simple Example (cont)

```
bar:
    foo.init();
    M1.mult(a, b);
    M2.mult(c, d);

M1:
    mult(int x, int y) {
        foo.add(x * y)
    }

M2:
    mult(int x, int y) {
        foo.add(x * y)
    }

foo:
    int sum;
    SYNC init() {
        sum = 0;
    }
    add(int val) {
        sum += val;
    }
```
Short Read Alignment

Host
processReads()

Seeds
nextRead(readID, read)

nextRead(readID, read)
Host
- processReads()

- nextRead()

Seeds
- nextRead(readID, read)

- nextSeed()

Filter
- numSeeds(readID, read, count)
- numCALs(readID, count)
- addCAL(readID, CAL)

RIT
- nextSeed(readID, seed)

- numSeeds()
Host
processReads()

Seeds
nextRead(readID, read)
numSeeds()

Filter
numSeeds(readID, read, count)
numCALs(readID, count)
addCAL(readID, CAL)

RIT
nextSeed(readID, seed)
numCALs()
addCAL()

SWDispatch
nextRead(readID, read)
nextCAL(readID, CAL)
Seeds
nextSeed(readID, seed)
nextSeed()

Filter
numSeeds(readID, read, count)
numCALs(readID, count)
addCAL(readID, CAL)
nextRead()
nextCAL()

RIT
nextSeed(readID, seed)

Host
processReads()
nextRead()

SWDispatch
nextRead(readID, read)
nextCAL(readID, CAL)
nextRead()
nextCAL()

Reference
nextCAL(readID, CAL)
nextRef()
nextRead()
nextCAL()
nextRead()
nextCAL()

Host
processReads()
nextRead()

Seeds
nextSeed(readID, seed)
nextSeed()

Filter
numSeeds(readID, read, count)
numCALs(readID, count)
addCAL(readID, CAL)
nextRead()
nextCAL()
Seeds
nextSeed(readID, seed)
nextSeed()

Filter
numSeeds(readID, read, count)
numCALs(readID, count)
addCAL(readID, CAL)
nextRead()
nextCAL()

RIT
nextSeed(readID, seed)

Host
processReads()
nextRead()

SWDispatch
nextRead(readID, read)
nextCAL(readID, CAL)

nextCAL()
nextRead()

numCALs()
addCAL()

numSeeds()

nextSeed()

Reference
nextCAL(readID, CAL)

nextCAL()
nextRead()

nextRef()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()

nextCAL()

nextRef()

nextRead()
Dynamic Objects

- **Automatic allocation**
  - Object is allocated when method with new ID is called
  - All calls with same ID go to the same object
  - Object deallocates itself when done

- **Implementation**
  - Pool of objects (static)
  - Object manager
    - Allocates and maps objects
    - Directs method calls to appropriate object
Parallelism with Dynamic Objects
Distributed Objects

- Partitioned memory arrays
  - Concurrent accesses
  - Increased data bandwidth

- Partitioning memory causes associated objects to be copied and distributed with partitions

- Method calls automatically directed to right partition
Summary

- FPGAs are great for data/compute intensive applications
  - Power efficient
  - Massively parallel

- We need a better programming model
  - Write applications using object model
  - Programmer describes the parallelism using dynamic and distributed objects
  - Compile to a distributed accelerator system
  - Compiler handles memory partitioning, object duplication and distribution, and communication between objects