Maxeler Apps
N-Body Simulation

MAXELER Technologies
MAXIMUM PERFORMANCE COMPUTING

May 2015

N-Body Simulation

- Simulates interaction between N particles under gravitational forces in space
- A particle's state is described by its:
    - position (x, y, z),
    - velocity (x, y, z) and
    - mass

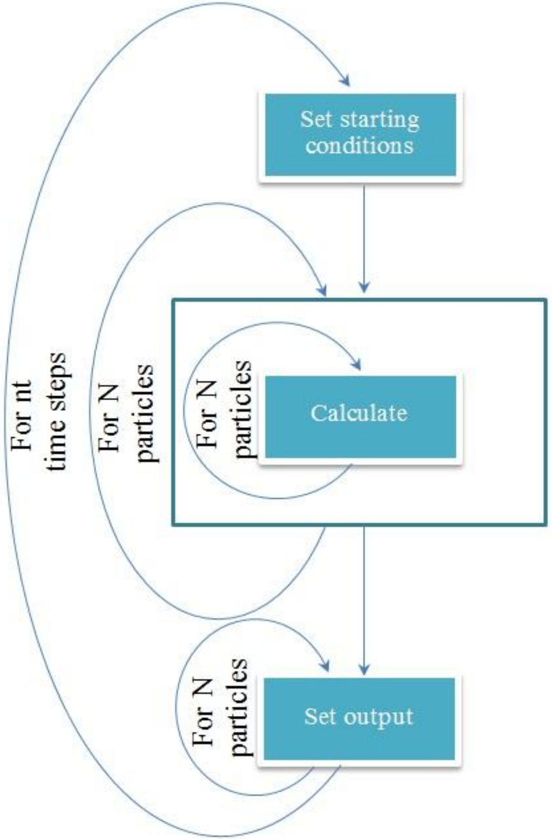
N-Body Method

- Particle interactions are described by partial differential equations
- The equations are solved numerically in order to obtain the state of the particles as a function of time

N-Body Parallelization Challenge

- Implementing a dynamic and irregular data structure in distributed memory
- The communication pattern depends on the input data and is unpredictable at compile time
- Minimizing communication time

Loop Flow Graph



N-Body Implementation on CPU

```
...
for (int t = 0; t < nt; t++) {                          // nt - Number of time-steps
    memset(acc, 0, N * sizeof(coord3d_t));              // N - Number of particles
    for (int q = 0; q < N; q++) {
      for (int j = 0; j < N; j++) {
          float rx = p[j].p.x - p[q].p.x;
          float ry = p[j].p.y - p[q].p.y;
          float rz = p[j].p.z - p[q].p.z;
          float dd = rx*rx + ry*ry + rz*rz + EPS;        // EPS - Damping factor
          float d = 1/ (dd*sqrtf(dd));
          float s = m[j] * d;                           // m - Masses of the N particles
          acc[q].x += rx * s;
          acc[q].y += ry * s;
          acc[q].z += rz * s;
      }
    }
    for (int i = 0; i < N; i++) {
      p[i].p.x += p[i].v.x;
      p[i].p.y += p[i].v.y;
      p[i].p.z += p[i].v.z;
      p[i].v.x += acc[i].x;
      p[i].v.y += acc[i].y;
      p[i].v.z += acc[i].z;
    }
  }
...
```

N-Body Implementation on DFE

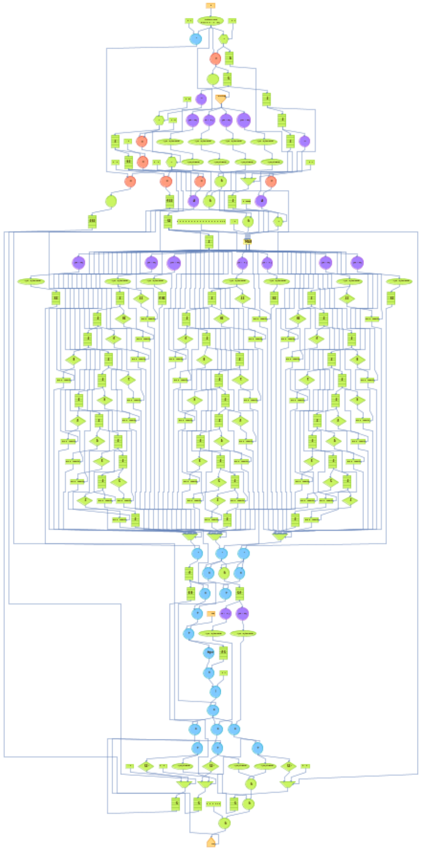
```
…
// all the below are interleaved data streams
DFEVar rx = pjX - piX;
DFEVar ry = pjY - piY;
DFEVar rz = pjZ - piZ;
DFEVar dd = rx*rx + ry*ry + rz*rz + scalars.EPS;
DFEVar d = 1 / (dd * KernelMath.sqrt(dd));
DFEVar s = pjM * d;
DFEParLoop lp = new DFEParLoop (this, “lp”);
lp.set_inputs(3, dfeFloat(8,24), 0.0);
DFEVar accX = lp.feedback[0] + rx*s;
DFEVar accY = lp.feedback[1] + ry*s;
DFEVar accZ = lp.feedback[2] + rz*s;
lp.set_outputs(accX, accY, accZ);
…
```

Resource Usage

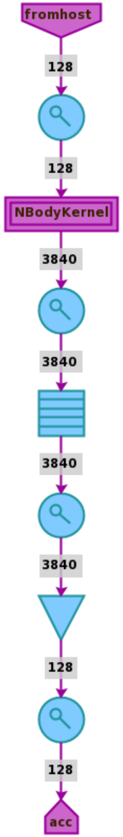
LUTs	FFs	BRAMs	DSPs	
297600	595200	1064	2016	total available resources for FPGA
161094	221592	336	1260	total resources used
54.13%	37.23%	31.58%	62.50%	% of available
153015	212697	272	1260	used by kernels
51.42%	35.74%	25.56%	62.50%	% of available
7984	8854	64	0	used by manager
2.68%	1.49%	6.02%	0.00%	% of available
45	41	0	0	stray resources
0.02%	0.01%	0.00%	0.00%	% of available

- For Vectis MAX3 Card

Simplified Kernel Graph



Manager Graph



N-Body Usage

```
Usage: NBody [OPTIONS]...

 -h, --help                  Print help and exit
 -V, --version               Print version and exit
 -N, --num-particles=INT     Maximum number of particles. The default value is
                             only used when the input is random.  (default='384')
 -t, --num-timesteps=INT     Number of time-steps  (default='1')
 -e, --EPS=FLOAT             Damping factor  (default='100')

Group: input
 -r, --random                Generate random input data
 -f, --file=FILE                   Read input data from file

Group: platform
 -c, --cpu                   Run the simulation on the CPU only
 -d, --dfe                   Run the simulation on the DFE only
 -m, --model                 Use the model when running the DFE
```

N-Body Sample Output

Command run:

```
./NBody -r -t 10 -N 64800
```

Output:

```
Running on DFE...

Wall clock time:                      14.0593       s
Run time:                             14.0505       s    (99.9%)
Update time:                          0.00881195    s    (0.1%)
DFE execution time:              14.1          s

Running on CPU…
CPU execution time:              388           s

Speed-up (1 card vs. 1 thread):  27.6x
Speed-up (node to node):         9.2x

Checking results...
PASSED
```

Conclusion

- The experimental results showed that there is a **significant speedup**, **near to thirty times**, in algorithm execution time when using DFE compared to the general purpose processor
- With the bigger input data size the speedup is expected to be increasing