

Hardware Implementation of Multigrid Algorithm



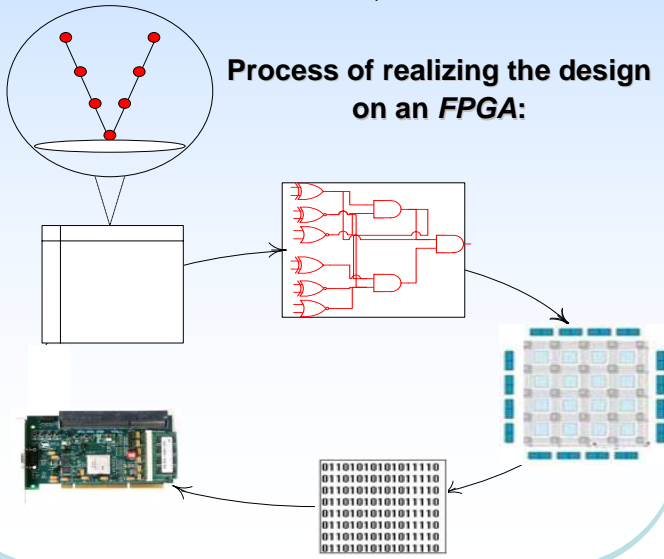
Safaa J. Kasbah and Issam W. Damaj



Over the past decades, Multigrid methods have been successfully applied to many problems, in diverse applications, such as computational fluid dynamics, material science, image processing and computer vision. Many efforts have been made to design and develop parallel Multigrid algorithms capable of solving very large size problems. However, the ideal performance of *MG* solvers is not accomplished yet due to many factors in practical-software- implementation. Reconfigurable Computing has shown significant progress in accelerating a wide variety of applications. This project explores possible hardware implementations of the V-cycle *MG* method for the solution to a 2-D Poisson equation. We draw a comparison between hardware implementation of *MG* and available software implementation.

Implementation

- *MG* methods are parallelizable in nature, Reconfigurable Computing exploits such Parallelism.
- Parallelizing *MG* components.
- Handel-C hardware compiler is used to code and implement our design and map it onto high-performance *FPGAs*: *Virtex II Pro*, *Altera Stratix*, and *Spartan3L* which is embedded in the *RC10 FPGA*-based system from *Celoxica*.
- Parallel execution of all simple statements where possible using the *Handel-C* construct '*par*'.
- The implementation performance is analyzed using the *FPGAs* vendors' tools: *Xilinx ISE*, *Quartus II*.



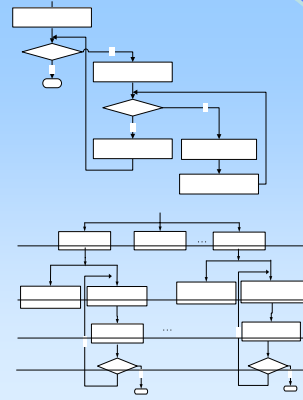
Example using '*par*' to parallelize *MG* Correct operator

```

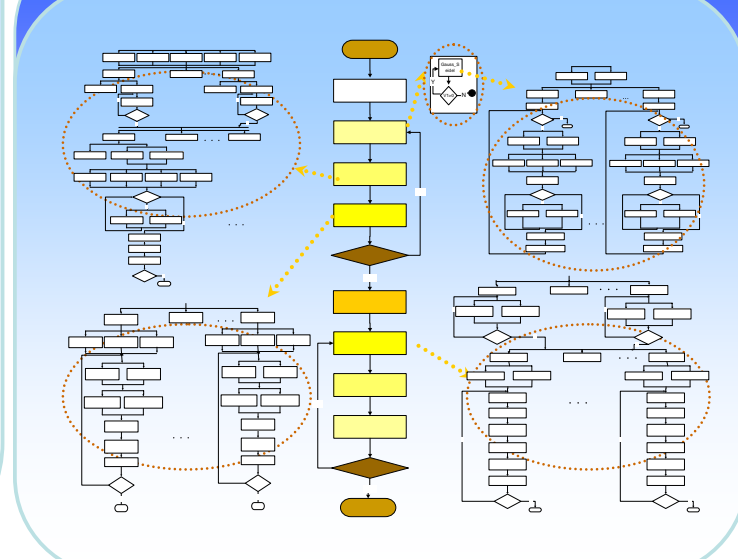
macro proc Correct()
{
for ( int i=1; i<=L; i++)
{
for ( int j=1; j<=L; j++)
psiNew[i][j] = FloatUnpackFromInt32(FloatPackInt32(psiNew[i][j])
+ FloatPackInt32(v[i][j]));
}
}
    
```

```

macro proc Correct()
{
i = 1;
par (i=1; i<=L; i++)
{
j = 1;
do
{
psiNew[i][j] = FloatUnpackFromInt32(FloatPackInt32(psiNew[i][j])
+ FloatPackInt32(v[i][j]));
j++;
} while(j<=L);
}
}
    
```

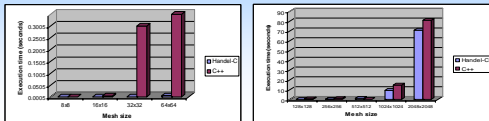


MG parallelization



Results

- Significant performance when compared to a software version running on a *GPP* and written in *C++*



Mesh Size	Execution Time (ms)	F _{max} (MHz)
8x8	0.000963	159.74
16x16	0.000208	153.52
32x32	0.001118	136.16
64x64	0.005558	115.97
128x128	0.031	83.91
256x256	0.188	64.60
512x512	1.308	31.46
1024x1024	9.3	17.60
2048x2048	70.97	9.28

- Synthesis Results:

Mesh Size	Execution Time (ms)	F _{max} (MHz)	Mesh Size	No. of Occupied Slices	Total equivalent gate count	Mesh Size	Total LE	LE Usage by No. of LUT Inputs	Total Registers
8x8	0.000963	159.74	8x8	264(5%)	5,990	8x8	725	402	228
16x16	0.000208	153.52	16x16	295(5%)	6,497	16x16	818	554	265
32x32	0.001118	136.16	32x32	415(8%)	9,321	32x32	925	625	301
64x64	0.005558	115.97	64x64	536(8%)	12,376	64x64	1068	709	360
128x128	0.031	83.91	128x128	798(16%)	18,107	128x128	1307	841	467
256x256	0.188	64.60	256x256	1,247(25%)	29,244	256x256	1739	1,070	670
512x512	1.308	31.46	512x512	2,125(43%)	51,115	512x512	2653	1,357	816
1024x1024	9.3	17.60	1024x1024	3,875(43%)	94,484	1024x1024	3491	1,809	1002
2048x2048	70.97	9.28	2048x2048	4,926(99%)	180,879	2048x2048	4501	2,201	1482

Spartan3L

Xilinx Virtex II Pro

Altera Stratix