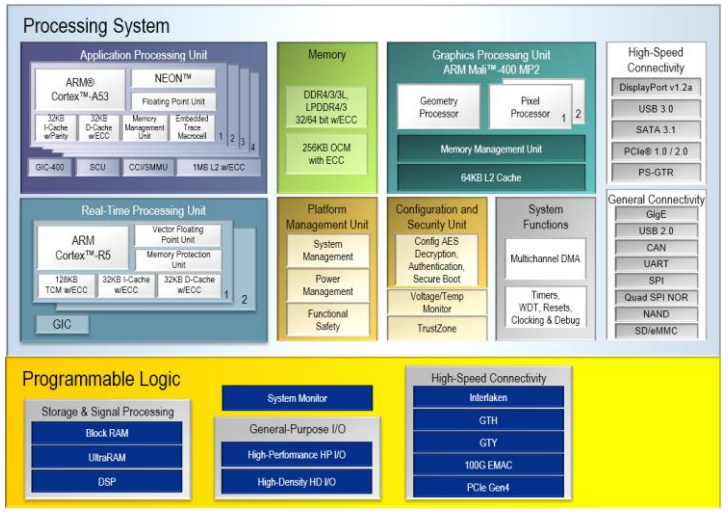
Sparse Matrix-Dense Matrix Multiplication on Heterogeneous CPU+FPGA Embedded System

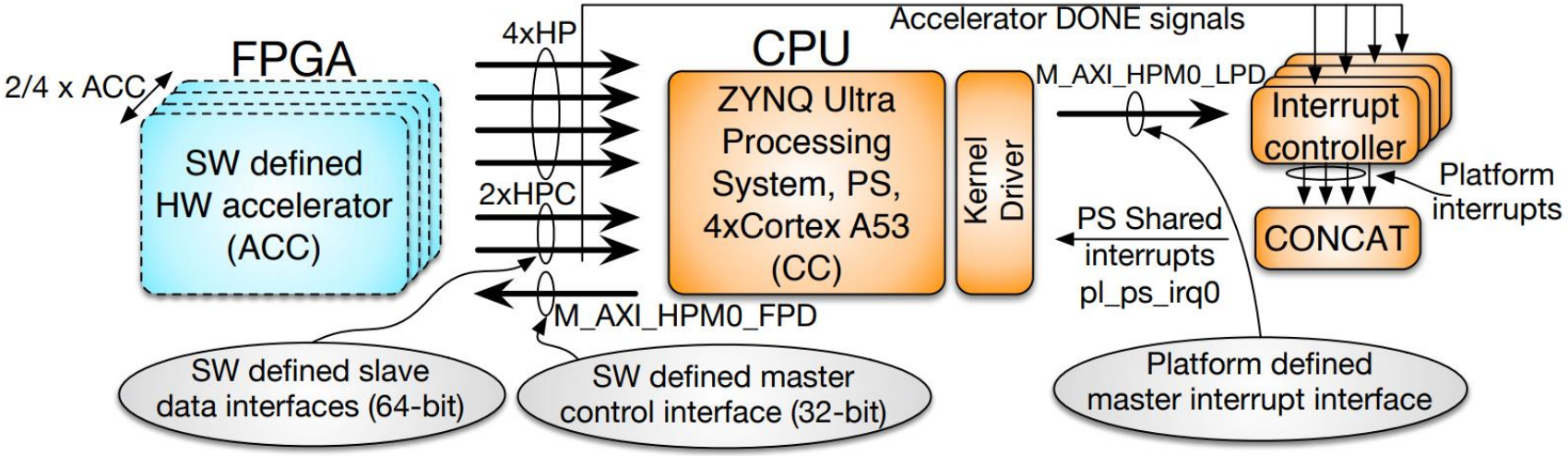
Mohammad Hosseinabady
Jose Nunez-Yanez
mohammad@hosseinabady.com
J.L.Nunez-Yanez@bristol.ac.uk
University of Bristol
Bristol

The Ultrascale+ MPSoC Platform and the ENEAC platform

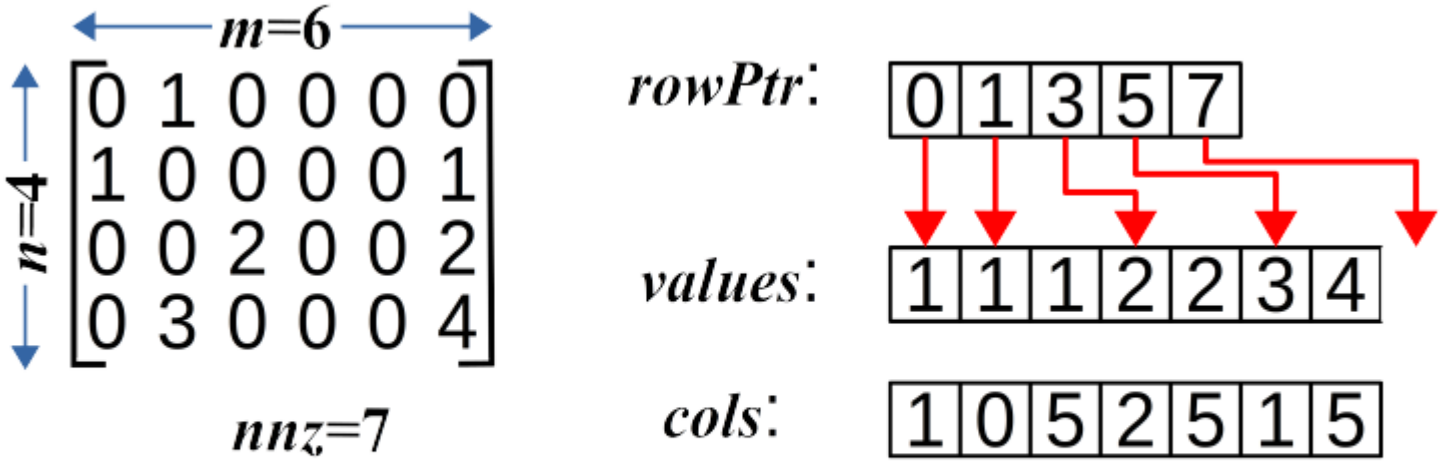
- 4C Cortex-A53 (**PS**)
- On-chip FPGA **(PL)**
- **PS** to **PL** via AXI
  - **2 HPC** and **4 HP** ports

System Logic Cells (K)	600
Memory (Mb)	32.1
DSP Slices	2,520
Maximum I/0 Pins	328





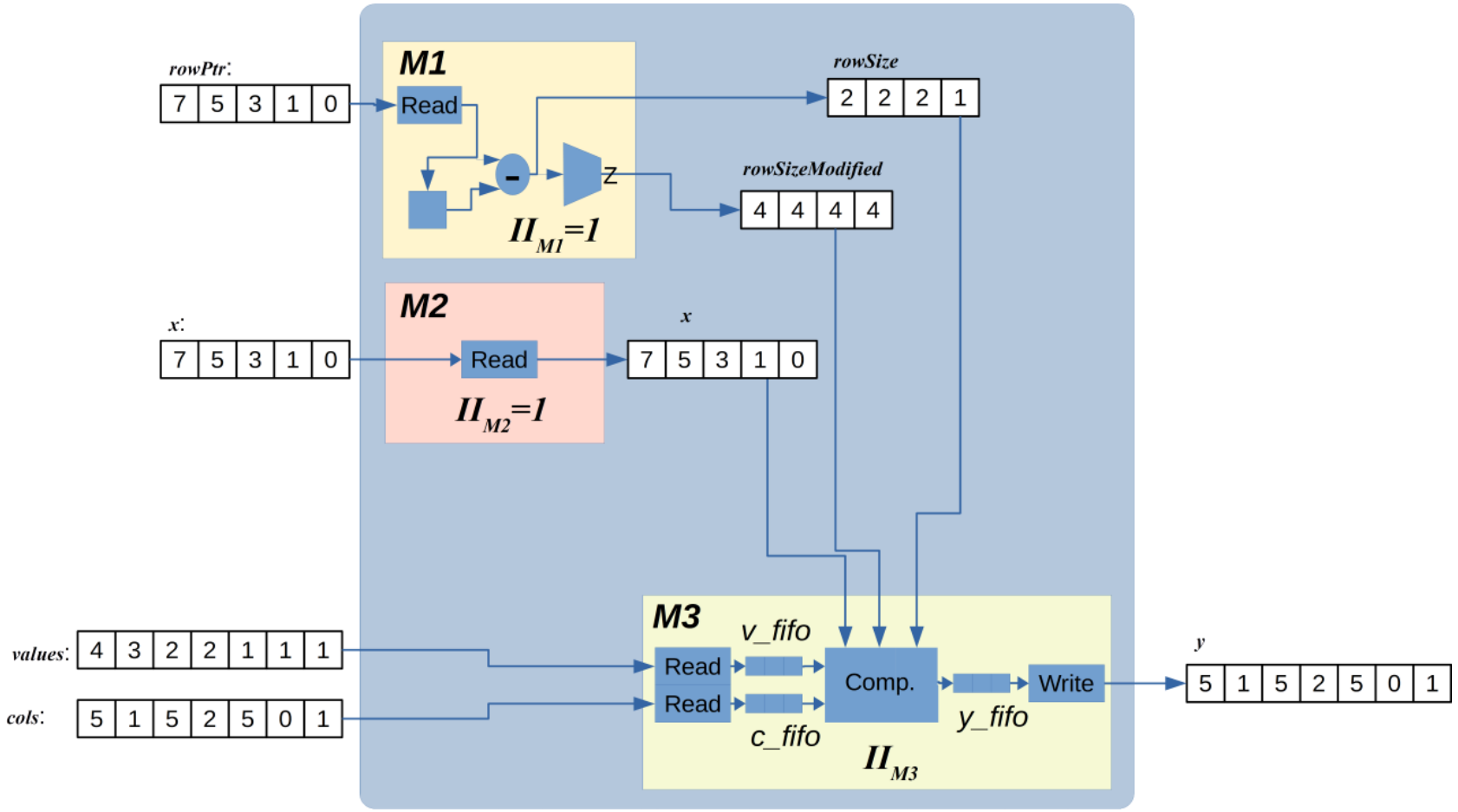
An example of a sparse matrix



a) An example **b) CSR format**

```
1  void SpMV (float *values, int *cols, int *rowPtr, float *x, float *y, int n) {
2     for (int i = 0; i < n; i+) {
3        float y0 = 0.0;
4        for (int j=rowPtr[i]; j<rowPtr[i+1]; j++){
5           y0 += value[j] * x[cols[j]]; {
6        y[i] = y0;
7     }
8  }
```

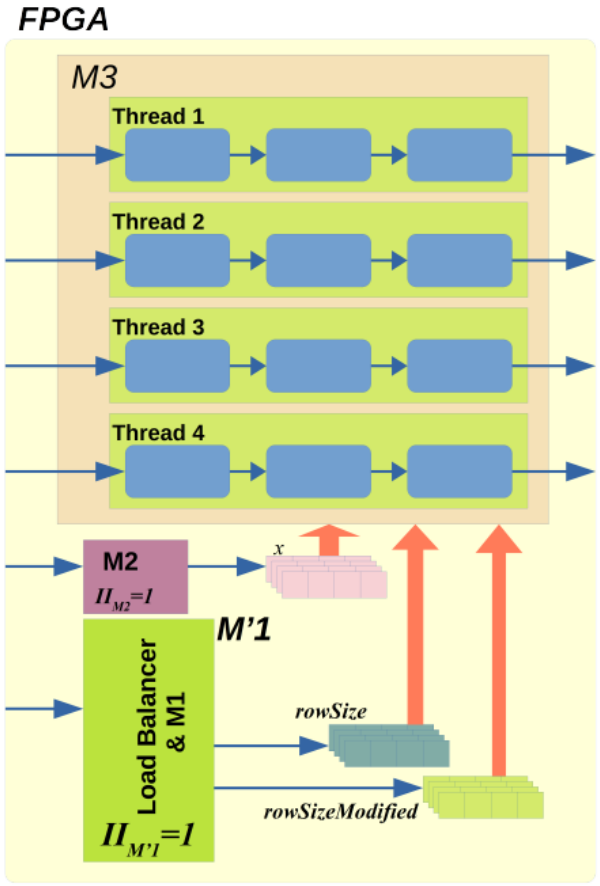
Streaming SpMV architecture



Hosseinabady, M., & Nunez-Yanez, J. (2019). A Streaming Dataflow Engine for Sparse Matrix-Vector Multiplication using High-Level Synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. https://doi.org/10.1109/TCAD.2019.2912923

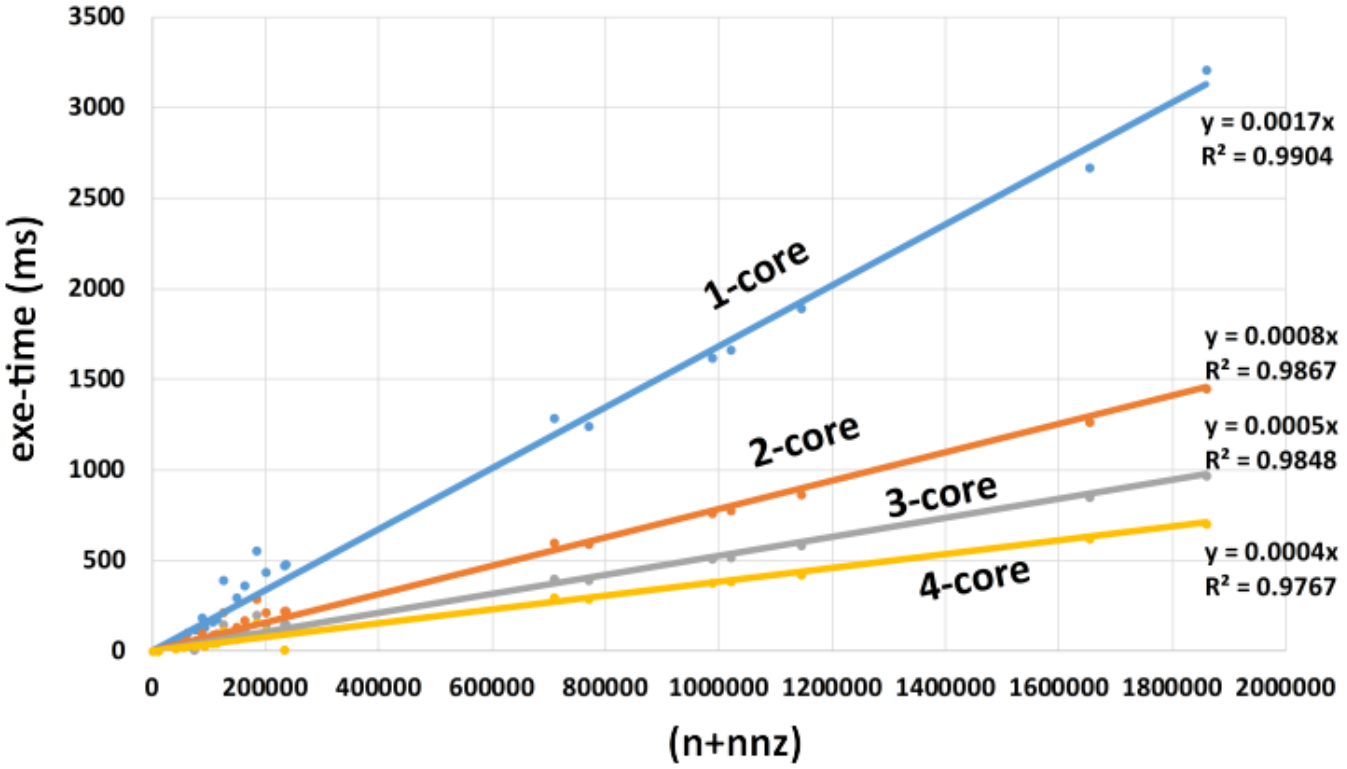
Stream computing code

```
1  u32 row_size_remains = 0;
2  for (u32 i = 0; i < new_nnz; i+=IIC {
3  #pragma HLS pipeline
4     if (row_size_tmp == 0) {
5        row_size_tmp = rowSizeModified[j];
6        row_size_remains = 0;
7        y_tmp = 0;
8        row_counter = rowSize [j++];
9     }
10    DATA_TYPE y_local = 0;
11    for (u32 p = 0; p < II; p++) {
12       row_size_remains++;
13       if (row_size_remains > row_counter) {
14          y_local += 0;
15       } else {
16          DATA_TYPE v = values_fifo.read();
17          u32 ci = col_indices_fifo.read ();
18          y_local += v*x_local[ci];
19       }
20    }
21    y_tmp += y_local;
22    row_size_tmp-=II;
23    if (row_size_tmp == 0) {
24       y_fifo << y_tmp;
25    }
26 }
```

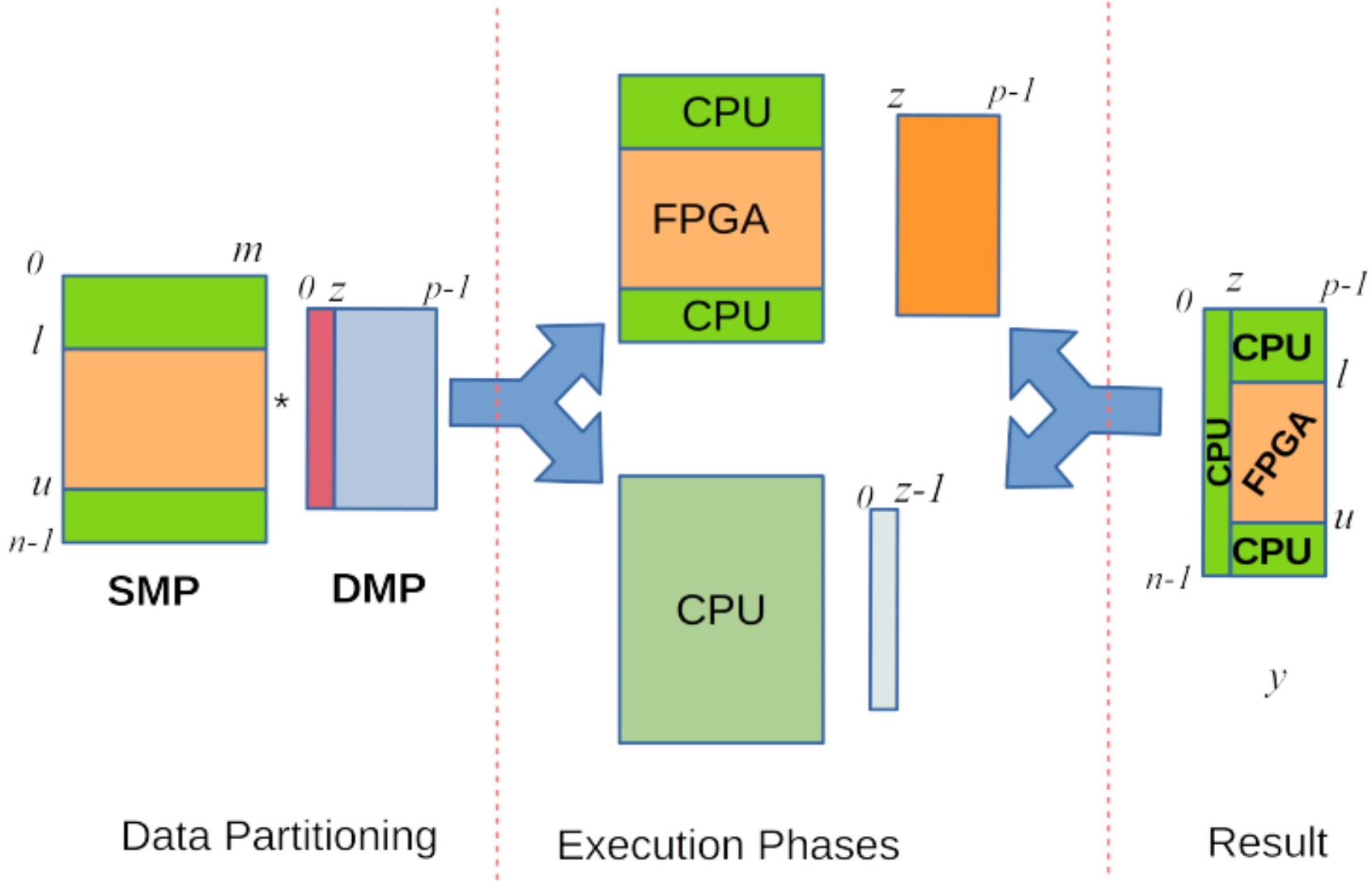


Multicore CPU implementation for SpMDM (p=100)

```
1  #pragma omp parallel for private(k, j, v)
2  for (v = 0; v < p; v++) {
3     for (i = 0; i < n; i++) {
4        y[v*m+i] = 0;
5        for (k = rowPtr[i]; k < rowPtr[i + 1]; k++) {
6           j = colIndices[k];
7           tmp = values[k] * x[v*m+j];
8           y[v*m+i] += tmp;
9        }
10    }
11 }
```



Sparse and dense matrix partitioning



Sparse matrix statistics

Matrix name	row	col	nnz	nnz-new
abtaha2	37932	332	137228	303456
bayer10	13436	13436	71509	138368
brack2	62631	62631	366559	867600
mixtank_new	29960	29960	1990385	2114272
c-45	13206	13206	93829	165088
opt1	15449	15449	973052	1028224
exdata_1	6001	6001	1137751	1168512

Heterogenous CPU+FPGA execution

Matrix name	3-core CPU	FPGA
abtaha2	112.7	50.93
bayer10	47.43	33.36
brack2	257.64	313.46
mixtank_new	1192.65	345.56
c-45	58.38	46.61
opt1	514.138	169.29
exdata_1	582.51	198.15

Matrix name	z (%)	l (%)	u (%)	exe-time(ms)	speed-up (%)
abtaha2	10	20	90	34.84	31.59
bayer10	35	20	100	22.55	32.40
brack2	55	10	100	170.5	33.82
mixtank_new	24	0	100	273	20.99
c-45	25	64	100	26.71	42.69
opt1	15	20	100	155.21	8.317
exdata_1	27	22	50	163.7	17.38

- Thanks to EPSRC and for the support with the ENEAC project.

- Questions ?