Binomni koeficijent MAXELER projekat

Profesor:
Veljko Milutinović
Autori:
Tatjana Damnjanović 223/2014
Katarina Đurić 236/2014

Definicija

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Binomna formula

$$(a \pm x)^n = a^n \pm n a^{n-1} x + \frac{n(n-1)}{2!} a^{n-2} x^2 \pm \frac{n(n-1)(n-2)}{3!} a^{n-3} x^3 + \cdots$$

$$= a^n \pm \binom{n}{1} a^{n-1} x + \binom{n}{2} a^{n-2} x^2 \pm \binom{n}{3} a^{n-3} x^3 + \cdots$$

Implementacija u programskom jeziku C

```
long long BK(int k,int n) {
        long long ans = 1;
        k = k > n-k ? n-k : k;
        int j = 1;
        for(; j <= k; j++, n--)
                if (n % j == 0)
                        ans *= n / j;
                else
                        if (ans % j == 0)
                                ans = ans / j*n;
                        else
                                ans = (ans*n) / j;
        return ans;
}
```

CPUCode/SimpleCpuCode.c

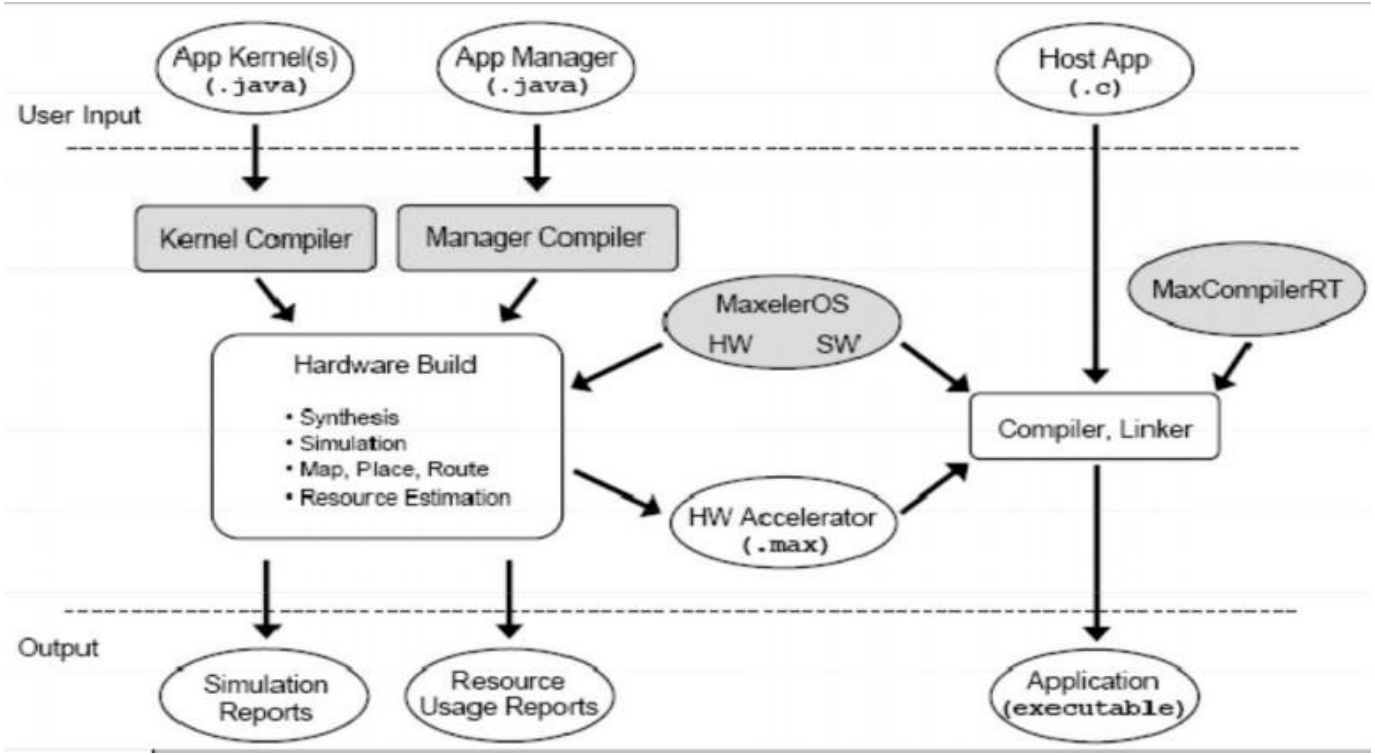
Save | Undo | Redo | Settings

```
#include <math.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
#include "Maxfiles.h"
#include <MaxSLiCInterface.h>

void generateInputData(float *dataIn, int size_k, int size_n) {
    int i=0;
    for (i = 0; i < size_k; i++)
        dataIn[i] = size_n - i;
}

long long BinCoefCPU(int k,int n) {
    long long ans = 1;
    k = k > n-k ? n-k : k;
    int j=1;
    for(;j<=k;j++,n--) {
        if(n%j==0)
            ans*=n/j;
        else
            if (ans % j == 0)
                ans=ans/j*n;
            else
                ans=(ans*n)/j;
    }
    return ans;
}
```

Maxeler


App Kernel(s)
(.java)
App Manager
(.java)
Host App
(.c)
User Input
Kernel Compiler
Manager Compiler
MaxelerOS
HW SW
MaxCompilerRT
Hardware Build
• Synthesis
• Simulation
• Map, Place, Route
• Resource Estimation
Compiler, Linker
HW Accelerator
(.max)
Output
Simulation Reports
Resource Usage Reports
Application
(executable)

BinCoefKernel

EngineCode/src/simple/SimpleKernel.maxj

Save | Undo | Redo | Settings

```
2
3   import com.maxeler.maxcompiler.v2.kernelcompiler.Kernel;
4   import com.maxeler.maxcompiler.v2.kernelcompiler.KernelParameters;
5   import com.maxeler.maxcompiler.v2.kernelcompiler.types.base.DFEVar;
6
7   class BinCoefKernel extends Kernel {
8       BinCoefKernel(KernelParameters parameters) {
9           super(parameters);
10
11          // Input
12          DFEVar x = io.input("x", dfeFloat(8, 24));
13          DFEVar count = control.count.simpleCounter(32);
14          DFEVar jedan = constant.var(dfeFloat(8,24), 1);
15
16          DFEVar result = x / (count.cast(dfeFloat(8,24)) + jedan) ;
17
18          // Output
19          io.output("y", result, dfeFloat(8, 24));
20      }
21  }
22
23
```

BinCoefManager

EngineCode/src/simple/SimpleManager.maxj

Save | Undo | Redo | Settings

```
1  package simple;
2
3  import com.maxeler.maxcompiler.v2.build.EngineParameters;
4  import com.maxeler.maxcompiler.v2.kernelcompiler.Kernel;
5  import com.maxeler.maxcompiler.v2.managers.standard.Manager;
6  import com.maxeler.maxcompiler.v2.managers.standard.Manager.IOType;
7
8  class SimpleManager {
9      public static void main(String[] args) {
10         EngineParameters params = new EngineParameters(args);
11         Manager manager = new Manager(params);
12         Kernel kernel = new SimpleKernel(manager.makeKernelParameters());
13         manager.setKernel(kernel);
14         manager.setIO(IOType.ALL_CPU);
15         manager.createSLiCinterface();
16         manager.build();
17     }
18 }
19
```

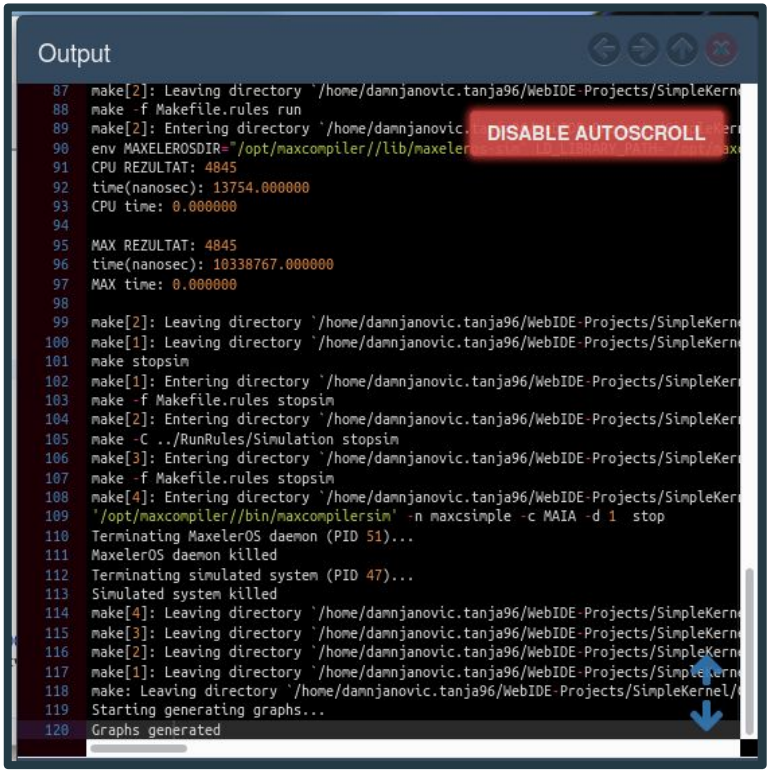
Main program

```
43
44  int main()
45  {
46      struct timespec start, stop;
47      clock_t begin;
48      clock_t end;
49      double result;
50      float timeInMillis;
51      const int size_k = 64;
52      const int size_n = 100;
53      float res = 1;
54      long long rezultat;
55      float *dataIn = malloc(size_k * sizeof(float));
56      float *dataOut = malloc(size_k*sizeof(float));
57
58      generateInputData(dataIn, size_k, size_n);
59
60      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start);
61      begin = clock();
62      rezultat = BinCoefCPU(size_k, size_n);
63      end = clock();
64      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &stop);
65
```

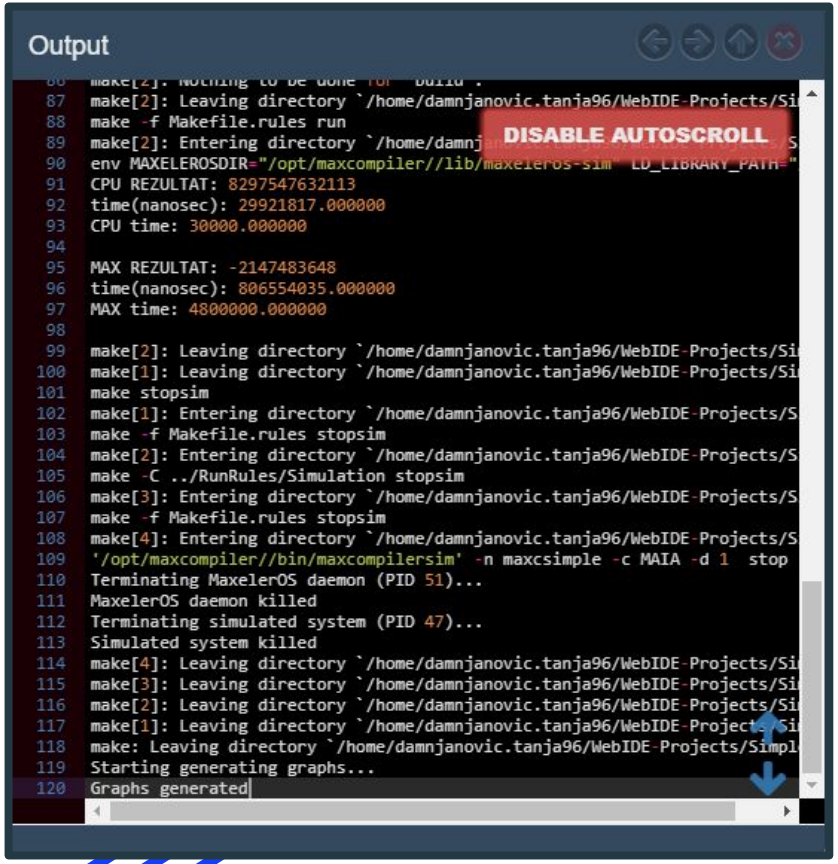
CPUCode/SimpleCpuCode.c

Save | Undo | Redo | Settings

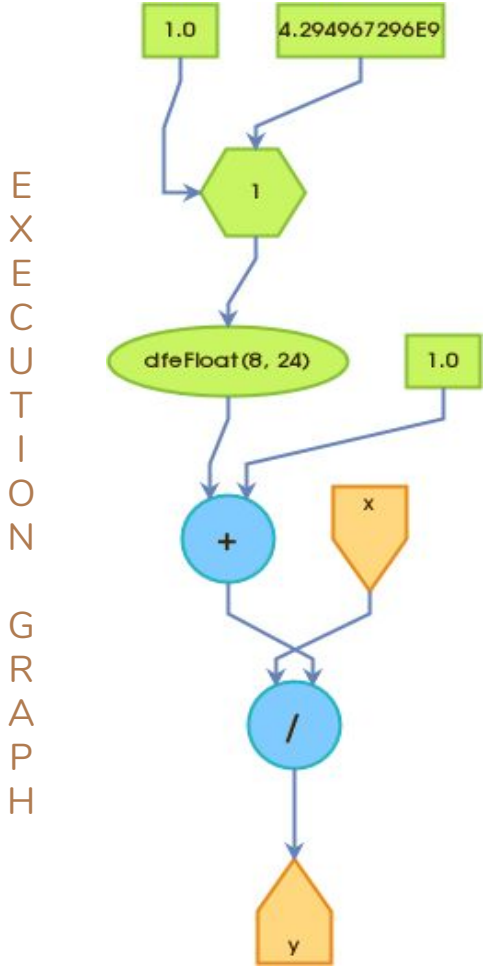
```
57
58      generateInputData(dataIn, size_k, size_n);
59
60      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start);
61      begin = clock();
62      rezultat = BinCoefCPU(size_k, size_n);
63      end = clock();
64      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &stop);
65
66      timeInMillis = (float)(end-begin)/(CLOCKS_PER_SEC/1000000);
67      result = (stop.tv_sec - start.tv_sec) * 1e6 + (stop.tv_nsec - start.tv_nsec);
68      printf("CPU REZULTAT: %lli\n", rezultat);
69      printf("time(nanosec): %lf\n", result);
70      printf("CPU time: %f\n\n", timeInMillis);
71
72      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start);
73      begin = clock();
74      BinCoef(size_k, dataIn, dataOut);
75      end = clock();
76      clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &stop);
77
78      for(int i = 0; i < size_k; i++)
79          res = res*dataOut[i];
80      timeInMillis = (float)(end-begin)/(CLOCKS_PER_SEC/1000000);
81      result = (stop.tv_sec - start.tv_sec) * 1e6 + (stop.tv_nsec - start.tv_nsec);
82      printf("MAX REZULTAT: %d\n", (int)round(res));
83      printf("time(nanosec): %lf\n", result);
84      printf("MAX time: %f\n\n", timeInMillis);
85
86  return 0;
```



Rezultat simulacije za ulaze manje od 100



Rezultat simulacije za ulaze veće od 1,000,000



ZAKLJUČAK

- Za ulaze ~100
  Maxeler je sporiji 750 puta
- Za ulaze ~10,000
  Maxeler je sporiji 68 puta
- Za ulaze ~1,000,000
  Maxeler je sporiji 27 puta

Failure is not the opposite of success;
It's part of it!

- Arianna Huffington

Kontakt

damnjanovic.tanja96@yahoo.com
kacaaa.30@gmail.com