Energy-Aware Compilation for Communicating Software Systems

Karel De Vogeleer - Télécom ParisTech / Département Informatique et Réseaux (INFRES)
Thesis: supervised by Gérard Memmi, Pierre Jouvelot & Fabien Coelho

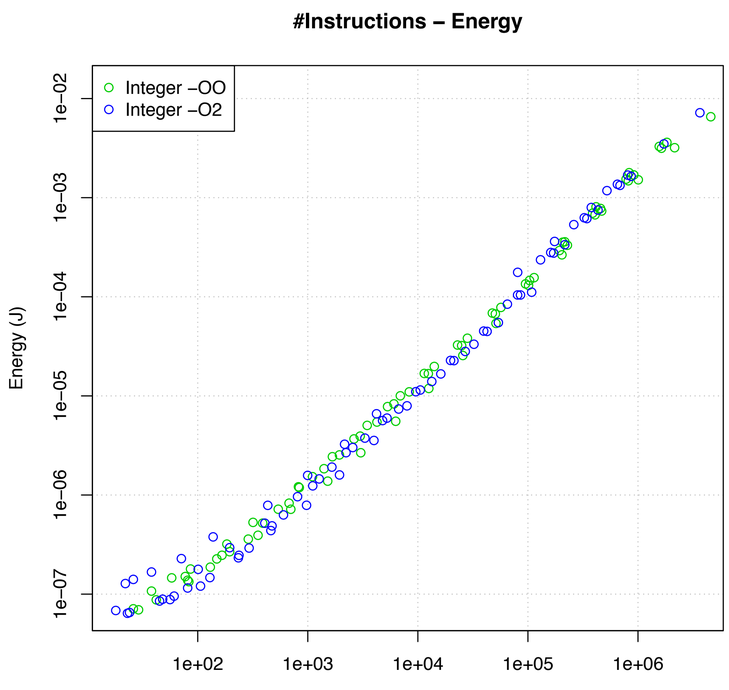
MOTIVATION

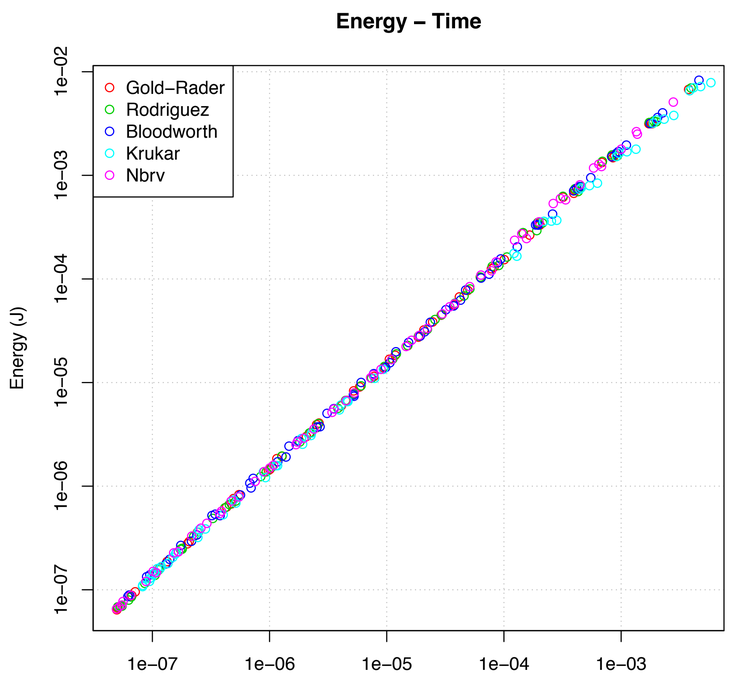
- The recent emergence of the Green-IT movement illustrates the energy-efficiency concern among providers of electrical communication devices. Even if computing consumes as a whole only a few percentage points of the world electrical energy, the significant increase in energy prices over the last few years drives this ecological trend. This is in sync with the economical aspect of leading companies to reduce their expenses. Moreover, on the consumer's side, with the advent of ubiquitous and mobile computing devices battery-life became an important user experience factor. Product and circuit design is also constrained by the peak-power heat dissipation of electrical components.

STRATEGY

- Hardware approaches to the energy consumption minimization problem within computing systems are not the only game in town. At a higher level, the hardware itself can also be used in an energy-efficient manner. Software algorithms may be designed (or refactored) and compiled in order to use minimal hardware resources, while proper energy management must be data-driven. This thesis strives to design compilation algorithms that generate code requiring a minimal amount of energy to execute, while preserving adequate performance in terms of computing time and memory usage.
We present an energy measurement platform we developed, together with preliminary data showing how various implementations of the bit-reverse algorithm impact consumption on a Samsung Galaxy SII smartphone.

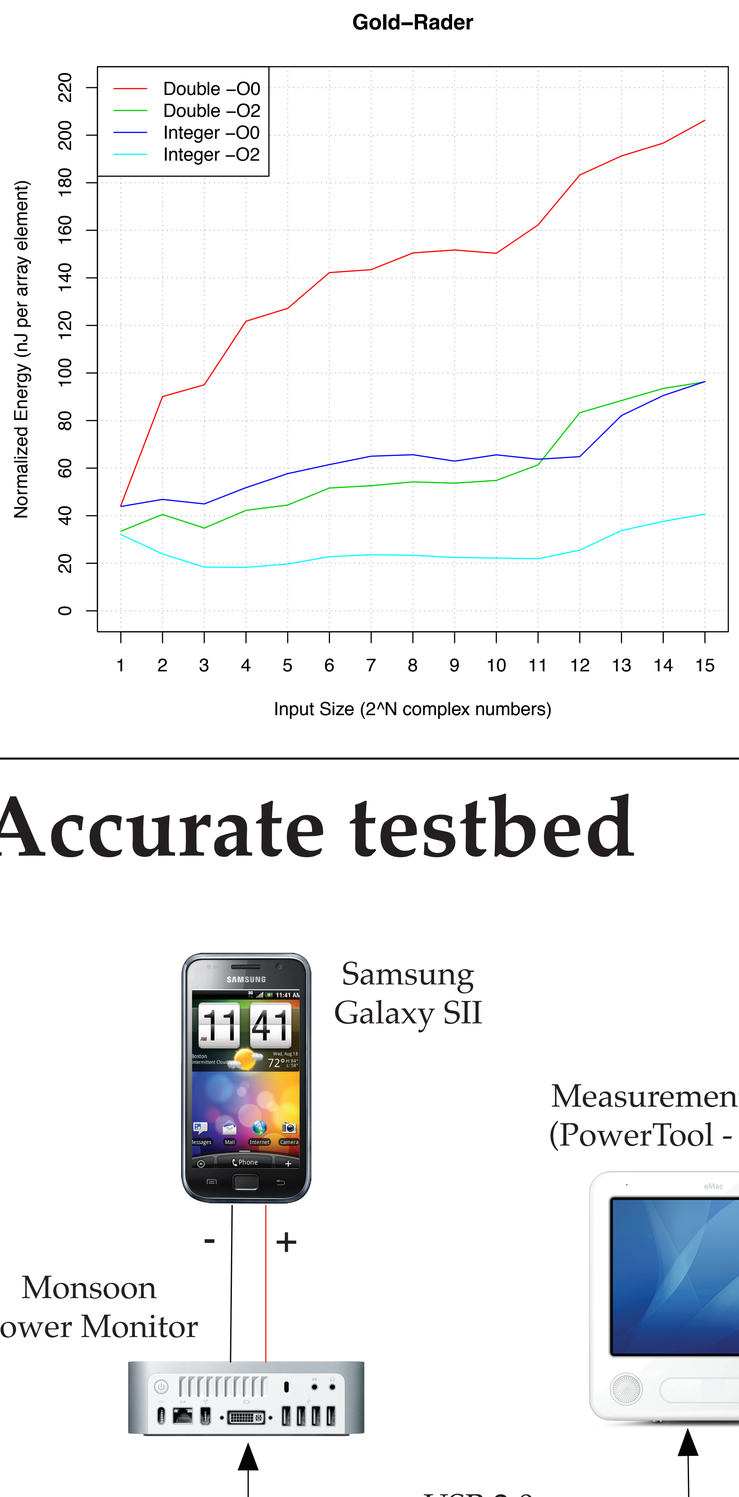
Bit-reverse for FFT: preliminary energy use measurements

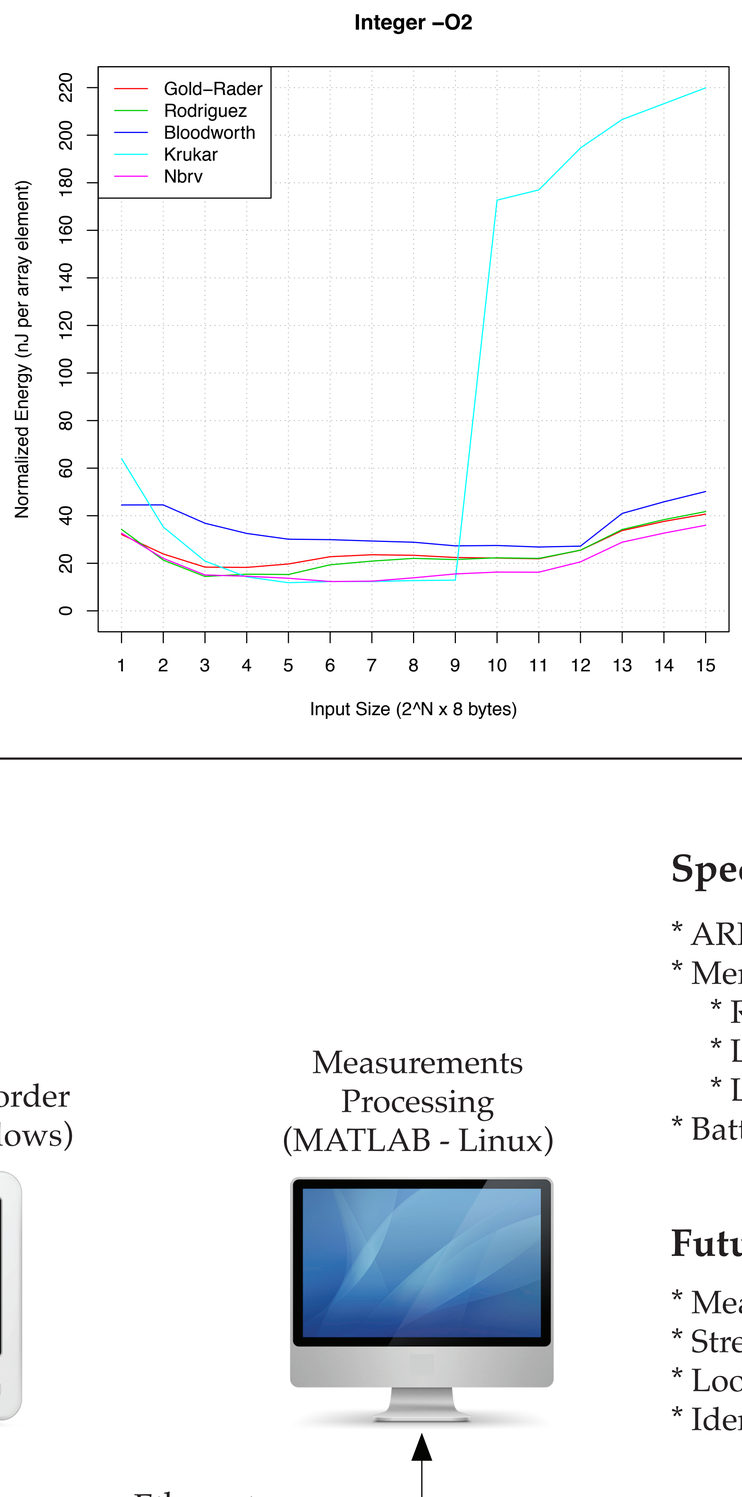





```
void bitreverse_gold_rader (int N, complex *data) {
   int n = N;
   int nm1 = n-1;
   int i = 0;
   int j = 0;

   for (; i < nm1; i++) { int k = n >> 1;
      if (i < j) {
         complex temp = data[i];
         data[i] = data[j];
         data[j] = temp;
      }
      while (k <= j) {
         j -= k;
         k >>= 1 ;
      }
      j += k ;
   }
}
```



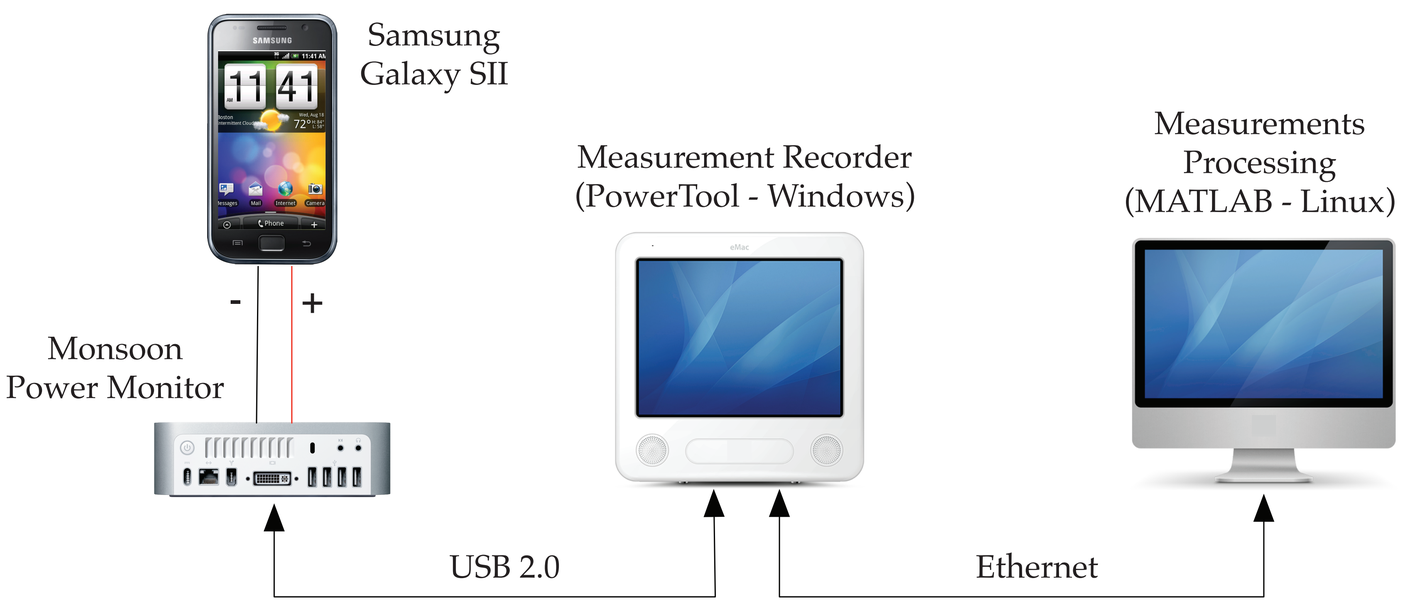



```
void bitreverse_krukar (int N, complex* complex_data) {
   int i, irev, j, bits;
   unsigned int temp, *ptr1, *ptr2;

   if(N <= 512) {
      bitreverse_krukar_512(N, complex_data); return;
   }

   unsigned int *data = (unsigned int*)complex_data;
   for(i = 0; i < N; i++) {
      irev = 0;
      for(j = 1, bits = (N >> 1);
            j < N;
            (j = j << 1), (bits = bits >> 1))
         if(i & j)
            irev |= bits;
         if(i < irev) {
            ptr1 = &(data[i << 1]);
            ptr2 = &(data[irev << 1]);
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
            ptr1++;
            ptr2++;
            temp = *ptr1;
            *ptr1 = *ptr2;
            *ptr2 = temp;
         }
      }
   }
```

Accurate testbed



Specifications Samsung Galaxy SII

* ARM Cortex A9 - Dual Core - 1.2 GHz
* Memory:
  * RAM: 1GB
  * L1: 32KB Instruction, 32KB Data; 4-way set-associative
  * L2: 1MB direct-mapped
* Battery: Li-ion 1650 mAh

Future Work

* Measuring larger programs
* Stressing all SoC parts
* Looking for memory limitations
* Identifying non-linear energy / time relationships