# An Introduction to Maxeler

Sasa Stojanovic
stojsasa@etf.rs
Veljko Milutinovic
vm@etf.rs

# Essence of the Maxeler Approach!

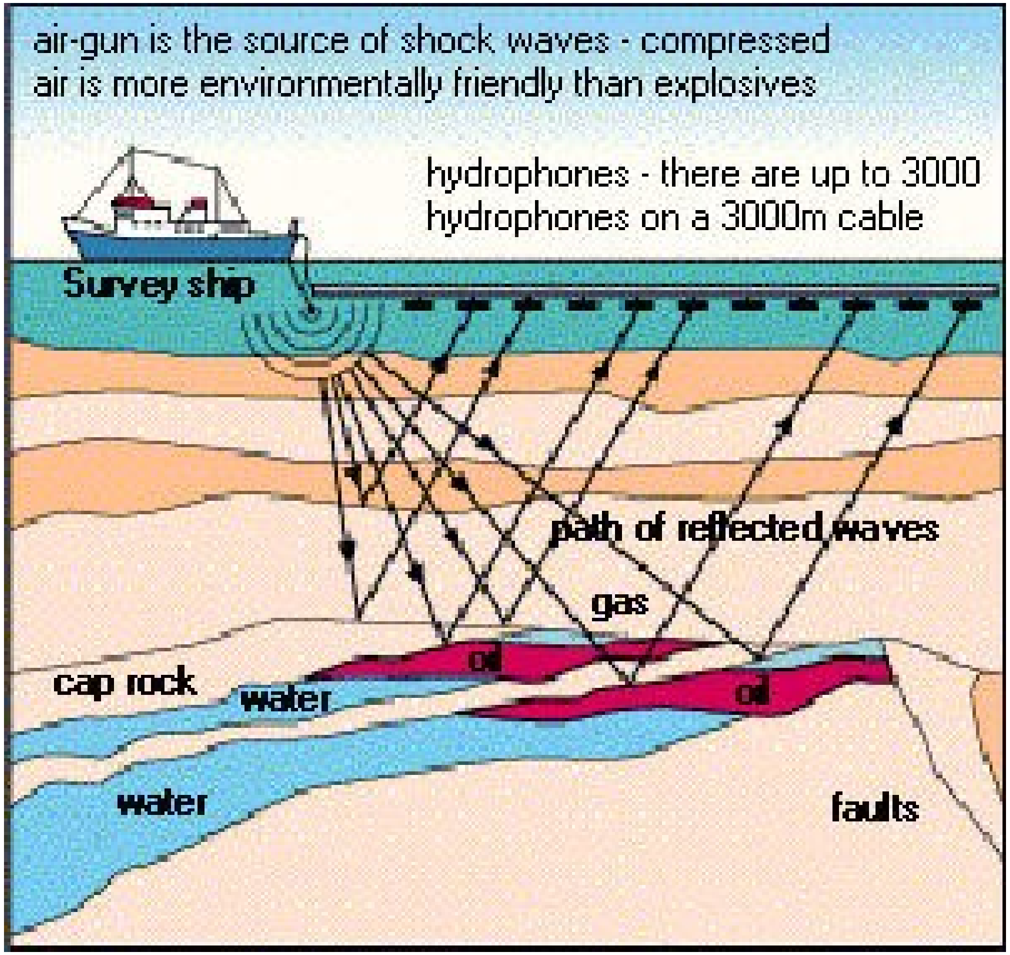
Compiling below the machine code level brings speedups;
also a smaller power, volume, and cost.

The price to pay:
The machine is more difficult to program.

Consequently:
Ideal for WORN applications :)

Examples:
GeoPhysisc, banking, dataminig in social networks, ...

# Examples of Maxeler Applications!


air-gun is the source of shock waves - compressed
air is more environmentally friendly than explosives
hydrophones - there are up to 3000
hydrophones on a 3000m cable
Survey ship
path of reflected waves
gas
oil
oil
cap rock
water
water
faults

# Examples of Maxeler Applications!

# Examples of Maxeler Applications!

# How-to? What-to?

One has to know
how to program Maxeler machines,
in order to get the best possible speedup out of them!

For some applications (G),
there is a large difference between
what an experienced programmer achieves,
and what an un-experienced one can achieve!

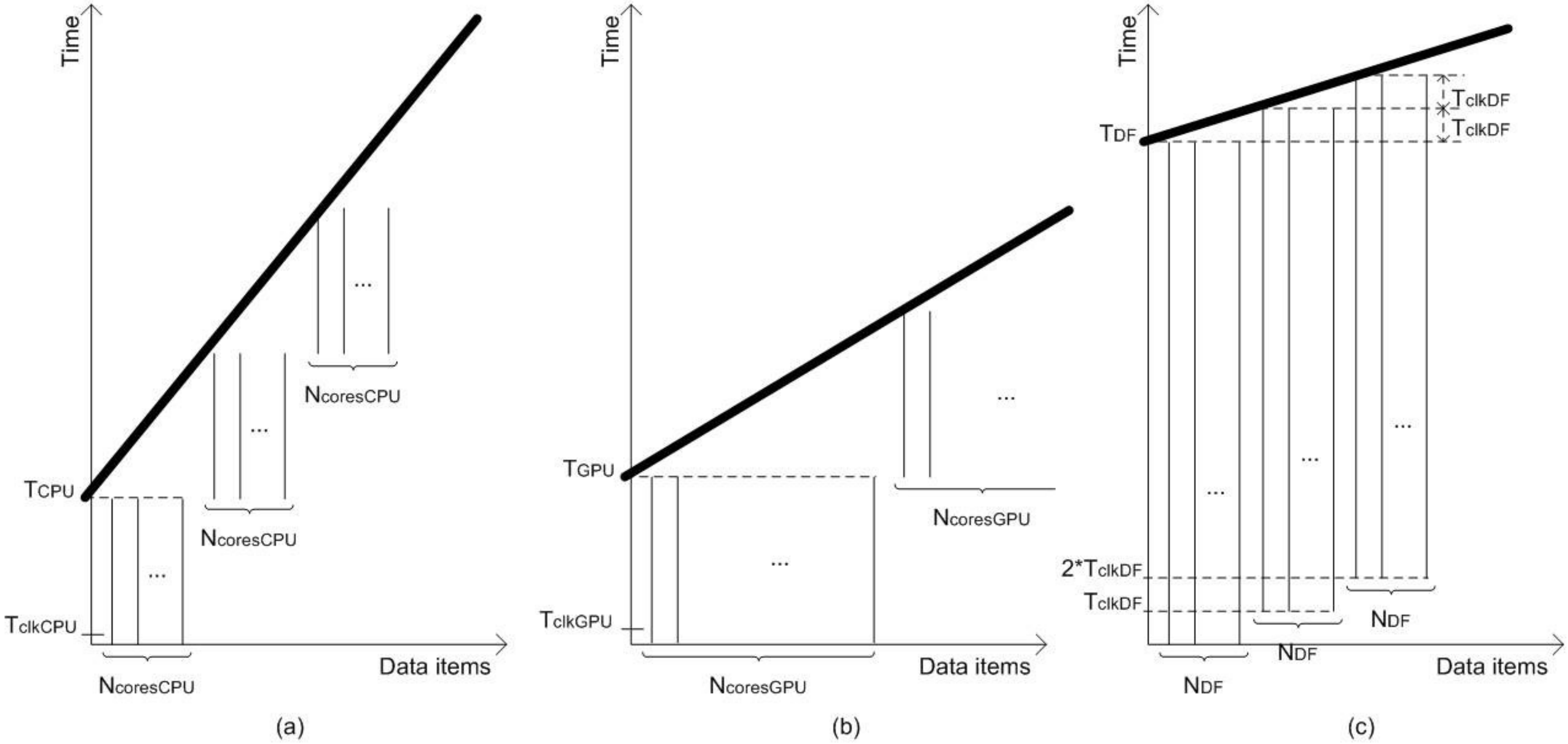
For some other applications (B),
no matter how experienced the programmer is,
the speedup will not be revolutionary
(may be even <1).

# The Essential Figure:

*tCPU =*
*N * NOPS * CCPU*TclkCPU*
*/NcoresCPU*

*tGPU =*
*N * NOPS * CGPU*TclkGPU /*
*NcoresGPU*

*tDF = NOPS * CDF * TclkDF +*
*(N – 1) * TclkDF / NDF*



Assumptions:

1. Software includes enough parallelism to keep all cores busy
2. The only limiting factor is the number of cores.

# Bottomline:

When is Maxeler better?
if the number of operations in a single loop iteration
    is above some critical value
then
    more data items means more advantage for Maxeler.

ADDITIVE SPEEDUP ENABLER

ADDITIVE SPEEDUP MAKER

In other words:
More data does not mean better performance
if the #operations/iteration is below a critical value.

Conclusion:
If we see an application with a small #operations/iteration,
  then
    it is possibly (not always) a “what-not-to” application,
    and we better execute it on the host;
else
    we will (or may) have a slowdown.

# To have it more concrete:

Maxeler: One new result in each cycle
  e.g.  Clock = 100MHz
        Period = 10ns
        One result every 10ns
[No matter how many operations in each loop iteration]

Consequently: More operations does not mean proportionally more time;
however, more operations means higher latency till the first result.

CPU: One new result after each iteration
  e.g. Clock=10GHz (!?)
       Period = 100ps
       One result every 100ps times #ops
[If #ops > 100 => Maxeler is better, although it uses a slower clock]

Also: The CPU example will feature an additional slowdown,
due to memory hierarchy access and pipeline related hazards
       =>
critical #ops (bringing the same performance) is significantly below 100!!!

# An Introduction to Maxeler

Q&A