Reconfigurable Systems

UAlg
UNIVERSIDADE DO ALGARVE

Jorge Semião

Instituto Superior de Engenharia

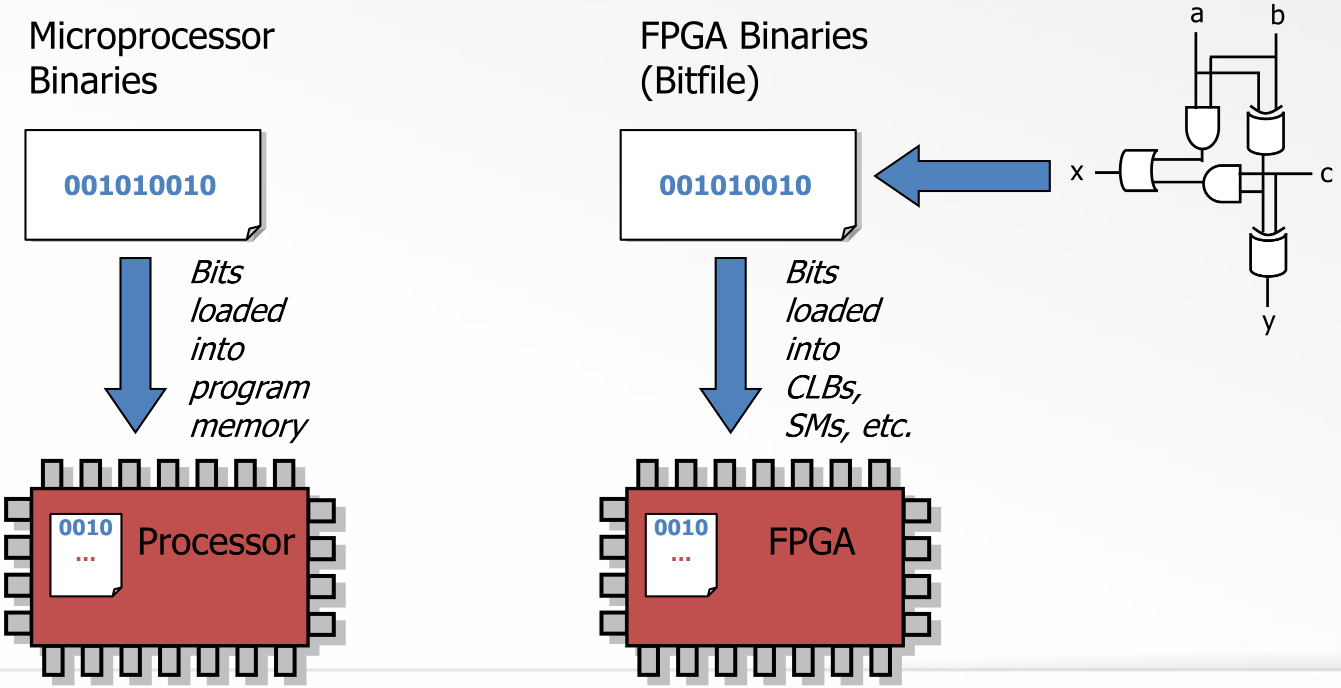
www.ualg.pt

Introduction to Reconfigurable Systems

- Can also be referred to Reconfigurable Computing
- Can be thought of as software-defined functionality, where flexibility is controlled predominately through the specification of bit patterns.
- Is the study of architectures that can adapt (after fabrication) to a specific application or application domain
  - Involves architecture, design strategies, tool flows, CAD, languages, algorithms, programmable hardware

What is Reconfigurable Computing?

- Alternatively, RS/RC is a way of implementing circuits without fabricating a device
  - Essentially allows circuits to be implemented as “software”
  - “circuits” are no longer the same thing as “hardware”
    - RC devices are programmable by downloading bits - just like software



Why is RS/RC important?

- Tremendous performance advantages
  - In some cases, > 100x faster than microprocessor
  - Alternatively, similar performances as large cluster
    - But smaller, lower power, cheaper, etc.
  - Example:

```
for (i=0; i < 16; i++)
   y += c[i] * x[i]
```

- Software executes sequentially
- RC executes all multiplications in parallel
  - Additions become tree of adders
- Even with slower clock, RC is likely much faster
- Performance difference even greater for larger input sizes
  - SW time increases linearly - O(n)
  - RC time is basically O(log2(n)) - If enough area is available

When to use RS/RC?

Implementation Possibilities

Microprocessor RC/RS (FPGA,CPLD, etc.) ASIC

Performance

Why not use an ASIC for everything?

Moore’s Law

- Moore's Law is the empirical observation made in 1965 that the number of transistors on an integrated circuit doubles every 18 months [Wikipedia]

- 1993: 1 Million transistors

2007: >1 BILLION transistors!!!!

Becoming extremely difficult to design this - ASICs are expensive!

Moore's Law

- Solution: Make billions of transistors into a reconfigurable fabric - fabricate 1 big chip and use it for many things
  - Area overhead: circuit in FPGA can require 20x more transistors
    - But, that's still equivalent to a > 50 million transistor ASIC
      - Pentium IV ~ 42 million transistors
  - Modern FPGAs reportedly support millions of logic gates!

2007: >1 BILLION transistors!!!!

Solution: Make this reconfigurable

When should RS/RC be used?

- 1) When it provides the cheapest solution
  - Depends on:
    - NRE Cost - Non-recurring engineering cost
      - Cost involved with designing system
    - Unit cost - cost of a manufacturing/purchasing a single device
    - Volume - # of units
  - Total cost = NRE + unit cost * volume
  - RC is typically more cost effective for low volume devices
    - RC: low NRE, high unit cost
    - ASIC: very high NRE, low unit cost

What about microprocessors?

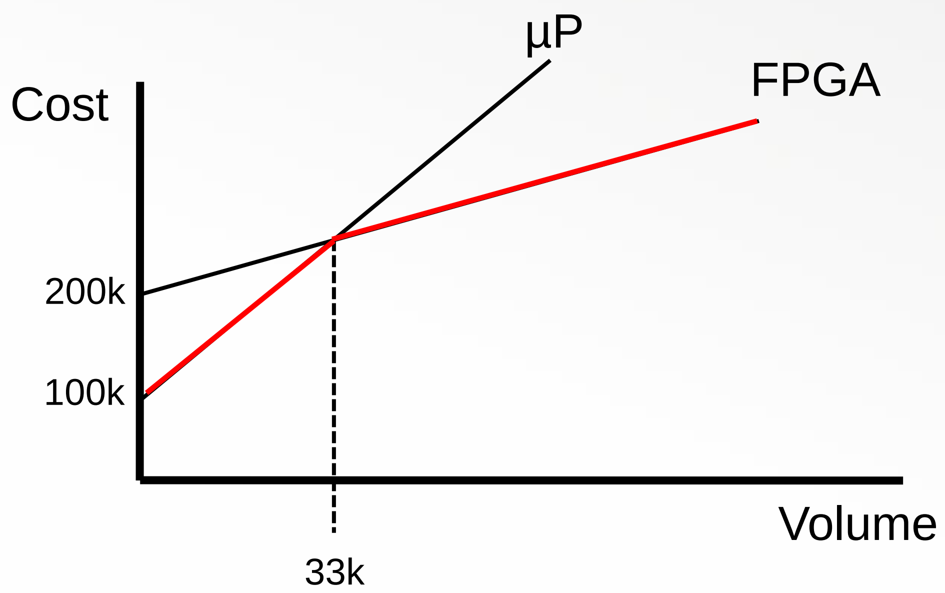
- Similar cost issues
  - uPs
    - low NRE cost (coding is cheap)
    - Unit cost varies from several dollars to several thousand
- Wouldn’t cheapest microprocessor always be the cheapest solution?
  - Yes, but …

What about microprocessors?

- Often, microprocessors cannot meet performance constraints
  - e.g. video decoder must achieve minimum frame rate
  - Common reason for using custom circuit implementation

Example

- FPGA: Unit cost = 5, NRE cost = 200,000
- Microprocessor (µP): Unit cost = 8, NRE cost = 100,000
- Problem: Find cheapest implementation for all possible volumes (assume both implementations meet constraints)



5v+200k = 8v+100k
v = 33k

Answer: For volumes less than 33k, µP is cheapest solution. For all other volumes, FPGA is cheapest solution.

Example: Your Turn

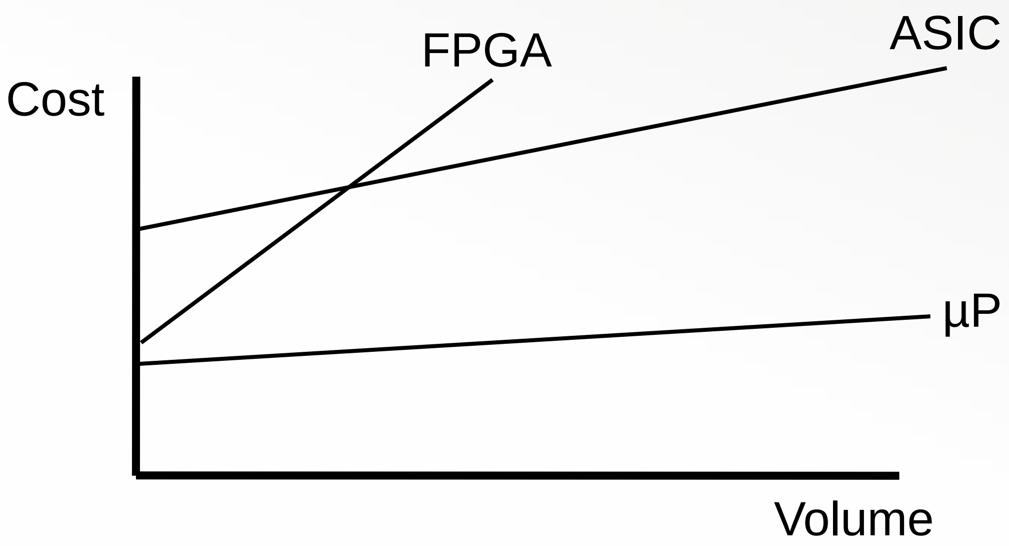
- FPGA
  - Unit cost: 6, NRE cost: 300,000
- ASIC
  - Unit cost: 2, NRE cost: 3,000,000
- Microprocessor (µP)
  - Unit cost: 10, NRE cost: 100,000
- Problem: Find cheapest implementation for all possible volumes (assume that all possibilities meet performance constraints)

Note: NRE meaning

Non-recurring Engineering. Non-recurring engineering cost refers to the one-time cost to research, design, develop and test a new product or product enhancement. When budgeting for a new product, NRE must be considered to analyze if a new product will be profitable. [Wikipedia]

Another Example

- FPGA
  - Unit cost: 7, NRE cost: 300,000
- ASIC
  - Unit cost: 4, NRE cost: 3,000,000
- Microprocessor (µP)
  - Unit cost: 1, NRE cost: 100,000

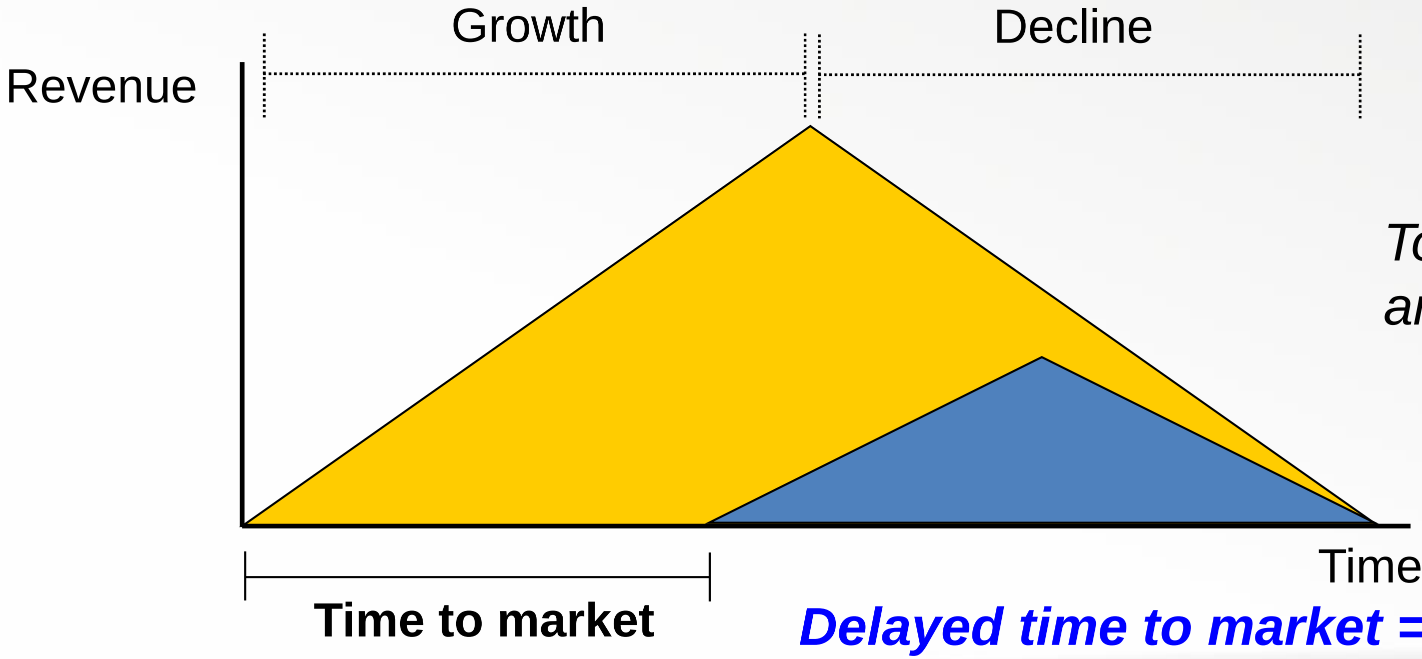


Answer: µP cheapest solution at any volume – not uncommon

When should RS/RC be used?

- 2) When time to market is critical
  - Huge effect on total revenue

RC has faster time to market than ASIC



Total revenue = area of triangle

Delayed time to market = less revenue

When should RS/RC be used?

- 3) When circuit may have to be modified
  - Can't change ASIC - hardware
  - Can change circuit implemented in FPGA
- Uses
  - When standards change
    - Codec changes after devices fabricated
  - Allows addition of new features to existing devices
  - Fault tolerance/recovery
  - "Partial reconfiguration" allows virtual fabric size - analogous to virtual memory
- Without RS/RC
  - Anything that may have to be reconfigured is implemented in software
    - Performance loss

Design Space Exploration

1. Determine architectures that meet performance requirements
   - Not trivial, requires performance analysis/estimation - important problem
     - Will study later in semester
   - And, other constraints - power, size, etc.
2. Estimate volume of device
3. Determine cheapest solution

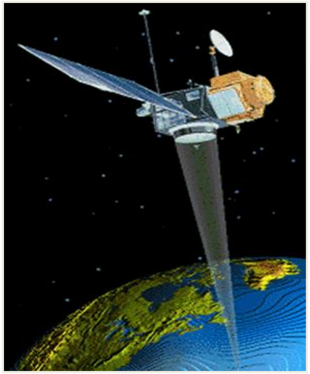
- The best architecture for an application is typically the cheapest one that meets all design constraints.

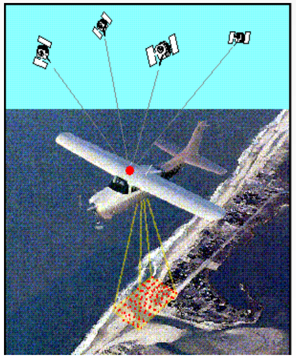
RS/RC Markets

- Embedded Systems
  - FPGAs appearing in set-top boxes, routers, audio equipment, etc.
  - Advantages
    - RC achieves performance close to ASIC, sometimes at much lower cost
      - Many other embedded systems still use ASIC due to high volume
        - » Cell phones, iPod, game consoles, etc.
    - Reconfigurable!
      - If standards changes, architecture is not fixed
      - Can add new features after production

RS/RC Markets

- High-performance embedded computing (HPEC)
  - High-performance/super computing with special needs (low power, low size/weight, etc.)
    - Satellite image processing
    - Target recognition
  - RS/RC Advantages
    - Much smaller/lower power than a supercomputer
    - Fault tolerance





RS/RC Markets

- High-performance computing - HPC
  - Cray XD-1
    - 12 AMD Opterons, FPGAs
  - SGI Altix
    - 64 Itaniums, FPGAs
  - IBM Chameleon
    - Cell processor, FPGAs
  - Many others
- RS/RC advantages
  - HPC used for many scientific apps
    - Low volume, ASIC rarely feasible

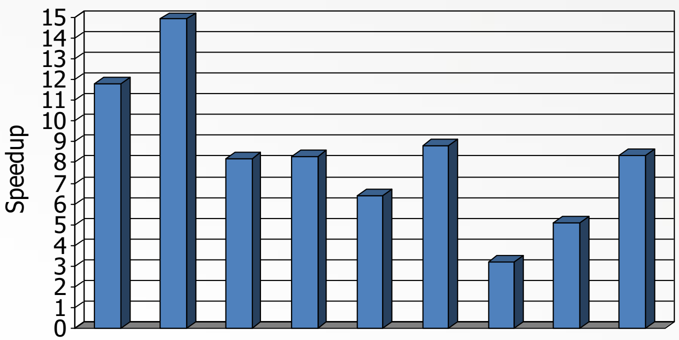
RS/RC Markets

- General-purpose computing???
  - Ideal situation: desktop machine/OS uses RC to speedup up all applications
  - Problems
    - RS/RC can be very fast, but not for all applications
      - Generally requires parallel algorithms
    - Coding constructs used in many applications not appropriate for hardware
  - Subject of tremendous amount of past and likely future research
- How to use extra transistors on general purpose CPUs?
  - More cache
  - More microprocessors
  - FPGA
  - Something else?

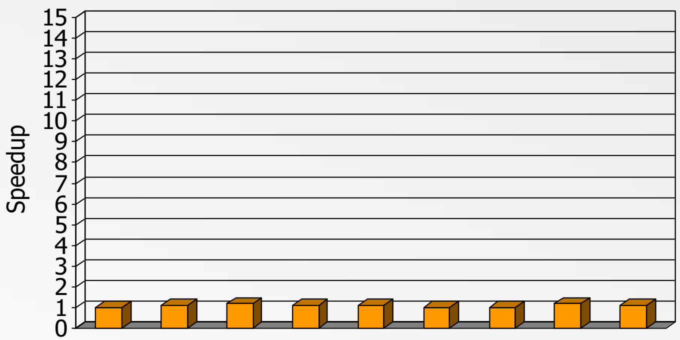
Limitations of RS/RC

- 1) Not all applications can be improved

Embedded Applications – Large Speedups



Desktop Applications – *No Speedup*



- 2) Tools need serious improvement!
- 3) Design strategies are often ad-hoc
- 4) Floating point?
  - Requires a lot of area, but becoming practical