EN2911X: Reconfigurable Computing

Lecture 01: Introduction

Prof. Sherief Reda

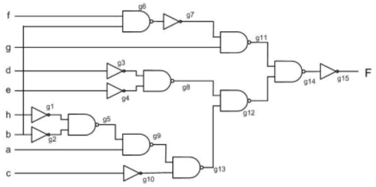
Division of Engineering, Brown University

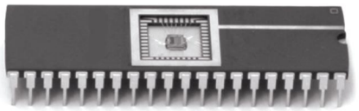
Fall 2009

BROWN

Methods for executing computations

Hardware (Application Specific Integrated Circuits)





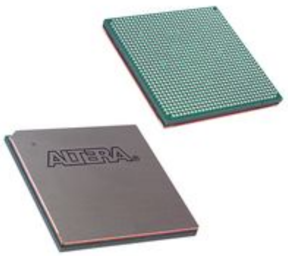
<u>Advantages:</u>

- very high performance and efficient

<u>Disadvantages:</u>

- not flexible (can’t be altered after fabrication)
- expensive

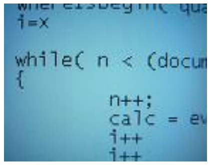
Reconfigurable computing



<u>Advantages:</u>

- fills the gap between hardware and software
- much higher performance than software
- higher level of flexibility than hardware

Software-programmed processors



```
whereIsBegin( qua
i=x

while( n < (docum
{
            n++;
            calc = ev
            i++
            i++
```

<u>Advantages:</u>

- software is very flexible to change

<u>Disadvantages:</u>

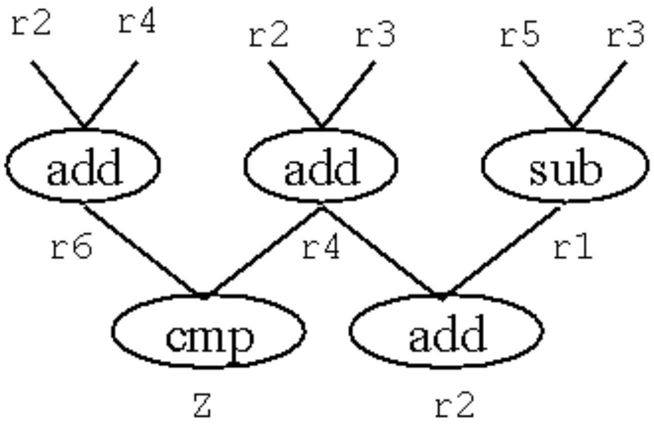
- performance can suffer if clock is not fast
- fixed instruction set by hardware

Temporal vs. spatial based computing

Temporal-based execution (software)

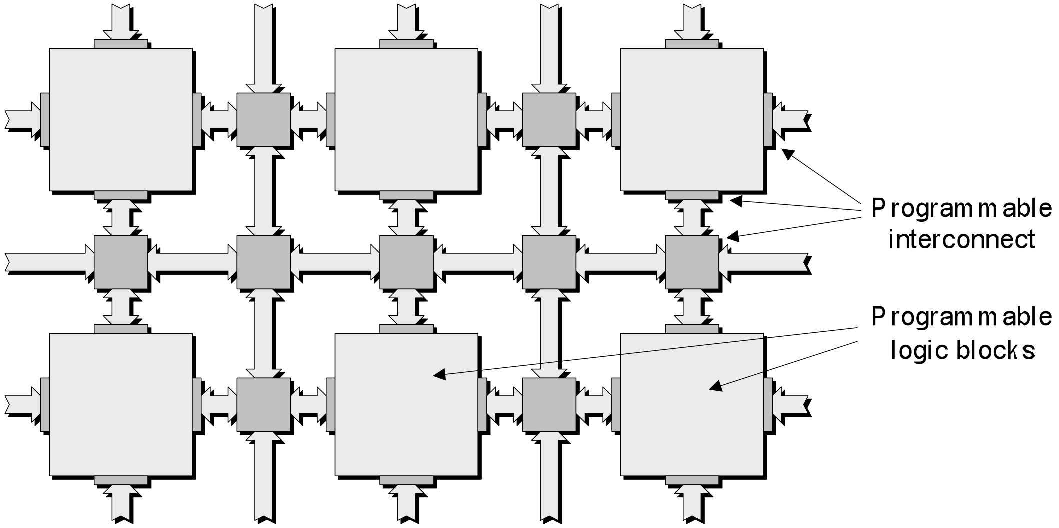
```
add    r2,r4,r6
add    r2,r3,r4
sub    r5,r3,r1
add    r4,r1,r2
cmp    r6,r4
```

Spatial-based execution (reconfigurable computing)



- ➢Ability to extract parallelism (or concurrency) from algorithmic descriptions is the key to acceleration using reconfigurable computing

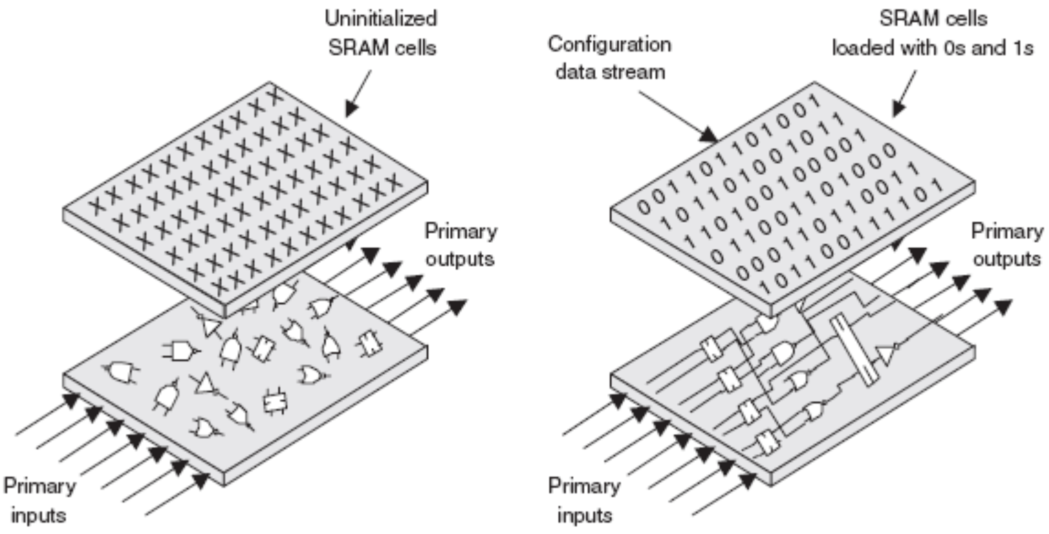
Reconfigurable devices



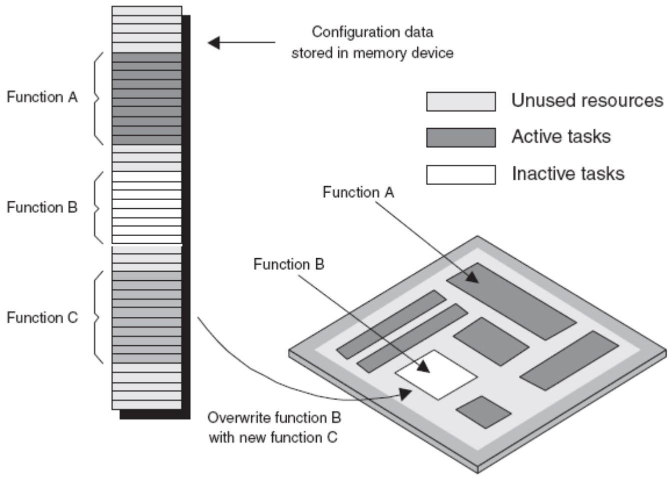
- *Field-Programmable Gate Arrays (FGPAs)* are one example of reconfigurable devices
- An FPGA consists of an array of *programmable logic blocks* whose functionality is determined by programmable configuration bits
- The logic blocks are connected by a set of *routing resources* that are also programmable

➢ Custom logic circuits can be *mapped* to the reconfigurable fabric

Configuring FPGAs



(a) Unconfigured (b) Configured



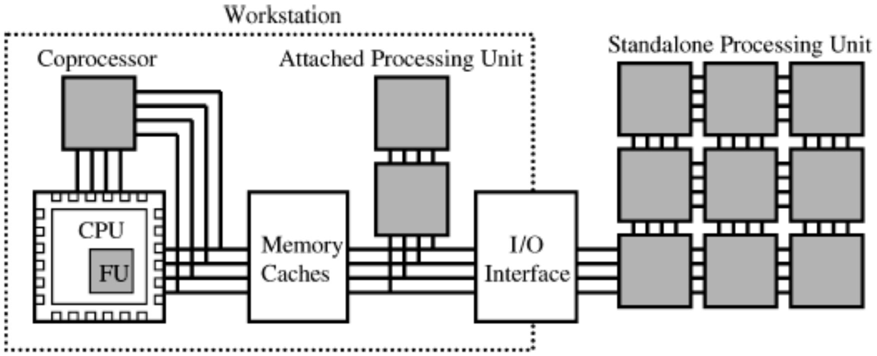
[Maxfield'04]

FPGAs can be dynamically reprogrammed before runtime or during runtime (virtual hardware)

- full
- partial

Uses of reconfigurable devices

1. Low/med volume IC production
2. Early prototyping and logic emulation
3. Accelerating algorithms in reconfigurable computing environments
   1. Reconfigurable functional units within a host processor (custom instructions)
   2. Reconfigurable units used as coprocessors
   3. Reconfigurable units that are accessed through external I/O or a network



[Compton'02]

Why reconfigurable computing is more relevant these days?

- Significant demand for high-performance computation. Large-scale data processing required in many areas (e.g. signal processing, particle physics, weather simulations, bioinformatics)
- *Why general-purpose processors are not meeting the demand?*
  - Single thread performance is no longer improving (individual core frequencies do not increase due to thermal problems).
  - Thread-level parallelism is not always attainable.
  - Consume large amount of power.
- *Why reconfigurable fabrics could meet the computational demand?*
  - Provide the spatial computational resources required to computationally process large streams of data directly in hardware
  - Economically feasible
  - Could lead to power savings

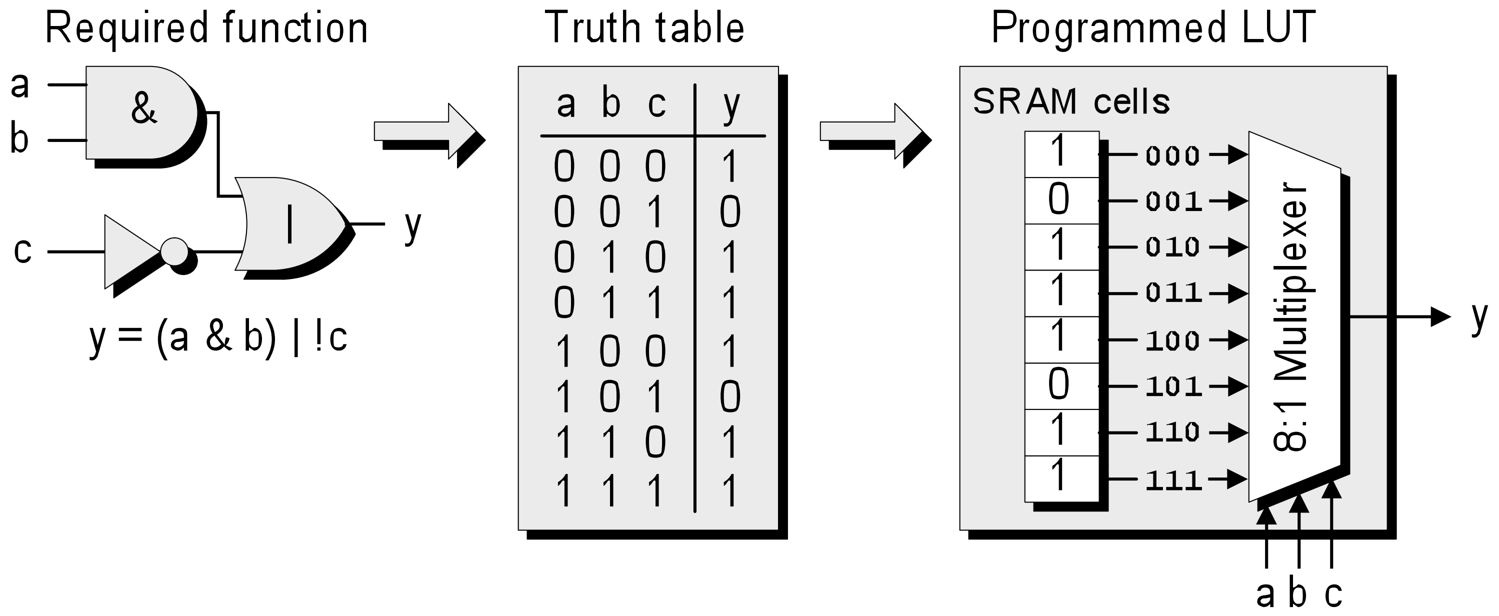
Goals of this class

1. Learn principles of reconfigurable computing.
2. Acquire hands-on experience and useful implementation skills for real-life engineering.
3. Develop/strengthen research skills through readings and class projects.

Topics covered in EN2911X

- Overview of programmable logic technology
- Rapid prototyping with FPGA boards
- Reconfigurable computing methodologies
- Hardware programming languages
- High-level system synthesis
- Soft multi-core system synthesis
- Algorithmic acceleration using reconfigurable computing
- Emerging reconfigurable systems

Topic 01: Programmable logic technology overview



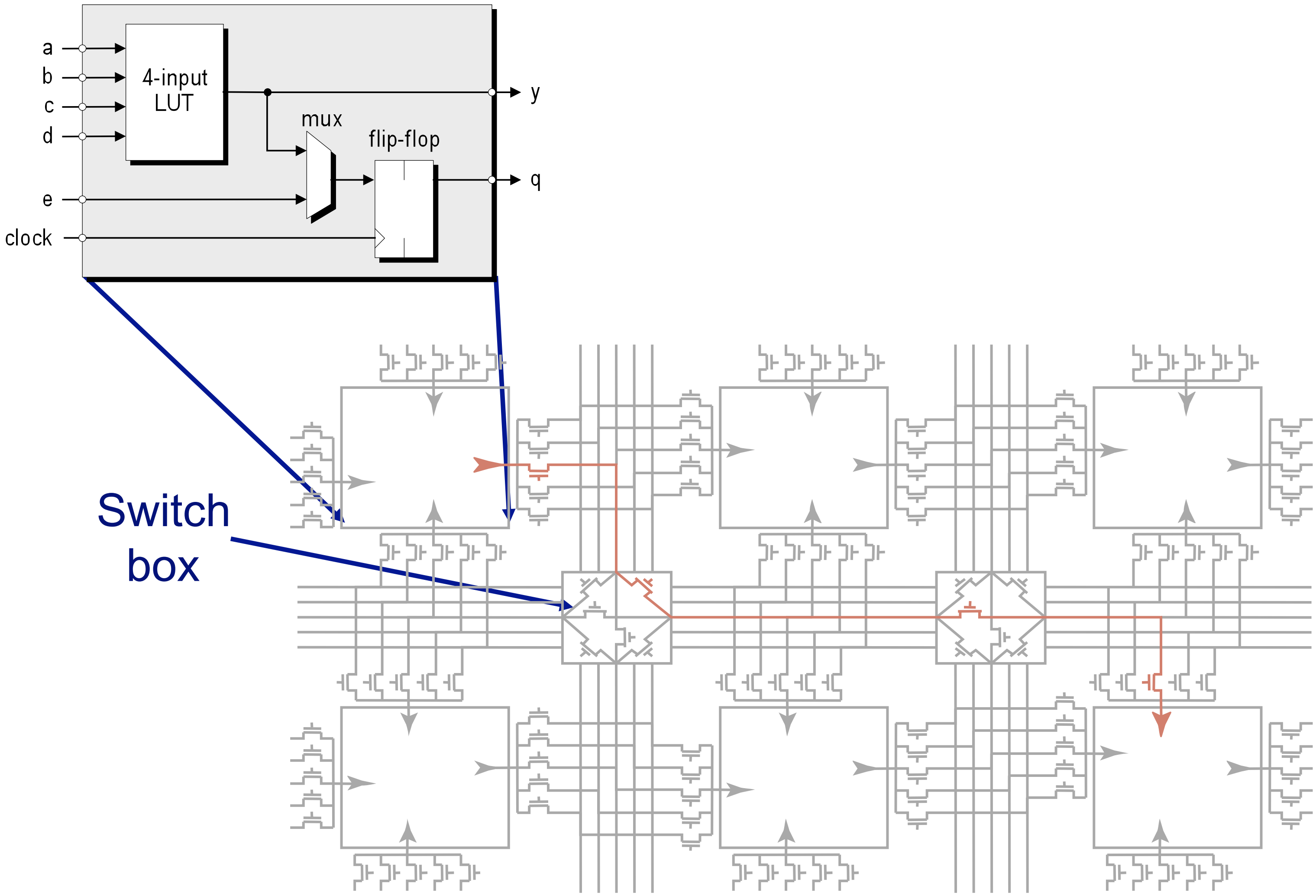
y = (a & b) | !c

a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Programming information could be stored in SRAM

4-input Look-Up Table (LUT) is the typical size

Topic 01: Programmable logic technology overview



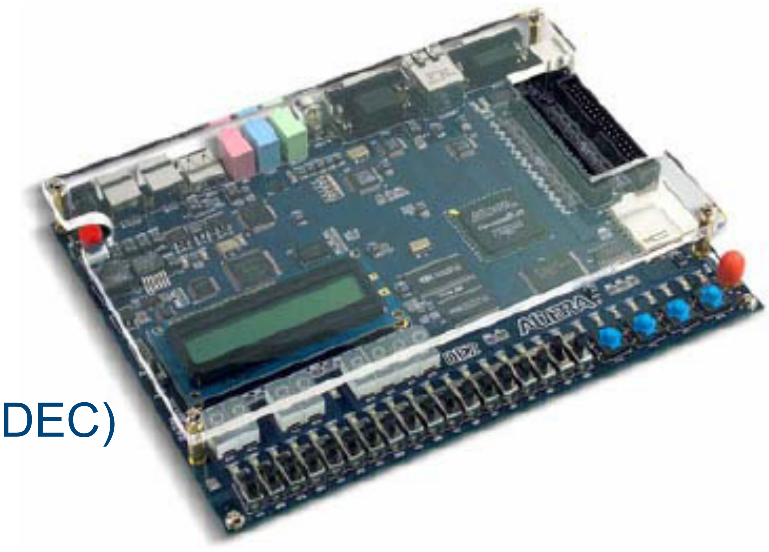
Switch box

Topic 02: Rapid prototyping with FPGA boards

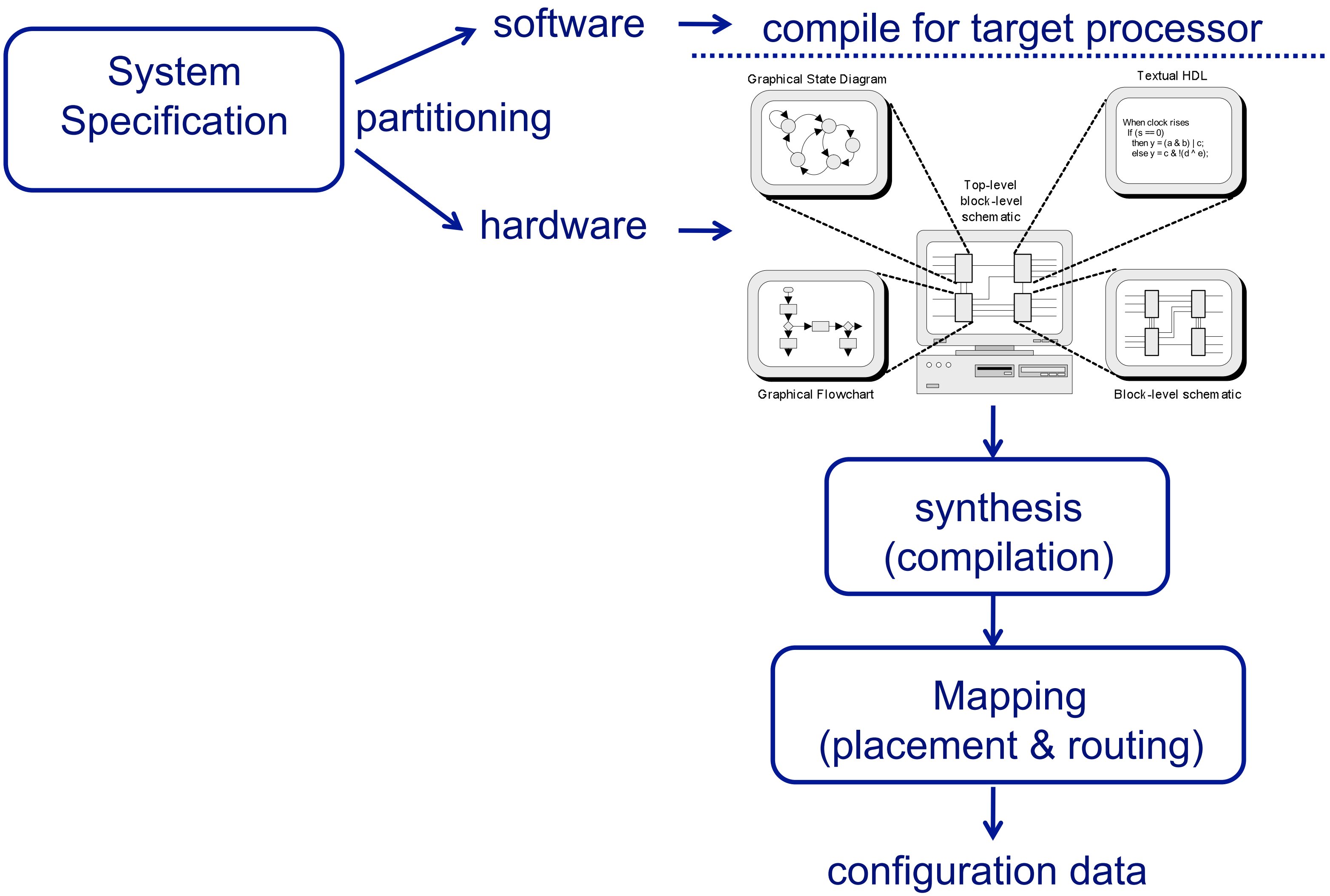
No need to design our board; we will use Altera's DE2 board and Quartus II software.

Features:

- Cyclone II FPGA 35K LUTs
- 10/100 Ethernet
- RS232
- Video out (VGA 10-bit DAC)
- Video in (NTSC/PAL/multi-format)
- USB 2.0 (type A and type B)
- PS/2 mouse or keyboard port
- Line in/out, microphone in (24-bit Audio CODEC)
- Expansion headers (76 signal pins)
- Infrared port
- Memory 8-MBytes SDRAM, 512K SRAM, 4-MBytes flash
- SD memory card slot
- Displays 16 x 2 LCD display
- Eight 7-segment displays
- Switches and LEDs



Topic 03: Reconfigurable computing methodologies



System Specification

partitioning

software → compile for target processor

hardware →

```
When clock rises
 If (s == 0)
  then y = (a & b) | c;
  else y = c & !(d ^ e);
```

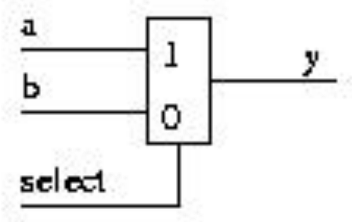
synthesis (compilation)

Mapping (placement & routing)

configuration data

Topic 04: Hardware programming languages (Verilog)

- Verilog is a hardware description language used to model digital systems
- Similar in syntax to C
- Differs from conventional programming languages as the execution of statements is not strictly linear. Possible to have sequential and concurrent execution statements
- The language can be synthesized into logic circuits



```
 module mux(a, b, select, y);
 input a, b, select;
 output y;
 initial
  begin
  always @ (a or b or select)
   if (select)
      y = a;
  else
      y = b;
  end
endmodule
```

Topic 05: High-level synthesis (MATLAB, C and SystemC)

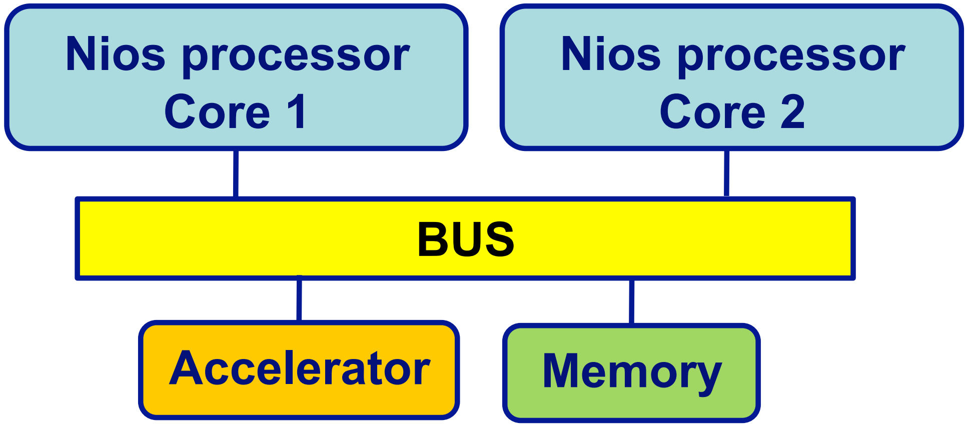
- SystemC is a system description language for hardware/software systems. SystemC is a set of library and macros implemented in C++ to allow specification and simulation of concurrent processes
- Direct C conversion is also getting main stream
- High-level synthesis from Matlab Simulink directly.

```
#include "systemc.h"
SC_MODULE(adder)
{
  sc_in<int> a, b;
  sc_out<int> sum;

  void do_add() {
    sum = a + b;
  }

  SC_CTOR(adder)  {
    SC_METHOD(do_add);
    sensitive << a << b;
  }
};
```

Topic 06: Algorithm acceleration using reconfigurable computing

- Learn how to use FPGAs and reconfigurable computing principles to accelerate algorithms: scientific computing, dynamic programming, combinatorial optimization, …, etc.
- Accelerating application in various fields:
  - Combinatorial algorithms (e.g., Boolean Satisfiability)
  - Signal and image processing (e.g., FFT)
  - Bioinformatics (e.g., DNA edit distance algorithm)
  - Pattern recognition (e.g., Viterbri algorithm)
  - …

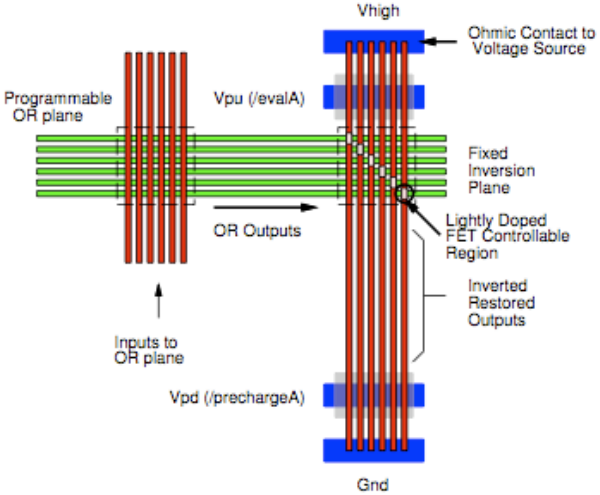
Topic 07: Soft multi-core computing environments



- Learn about hard and soft processors
- Design multi-core-based reconfigurable computing systems
- Design of on-chip networks for multi-core systems
- Design of custom instructions
- Design of pluggable acceleration function units

Topic 08: Emerging reconfigurable systems

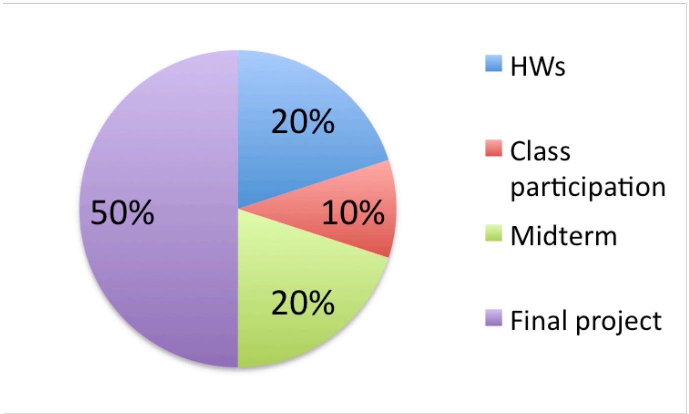
- Asynchronous FPGAs
- NanoPLA architectures
- 3D FPGAs
- MRAM FPGAs
- CNT-based FPGAs



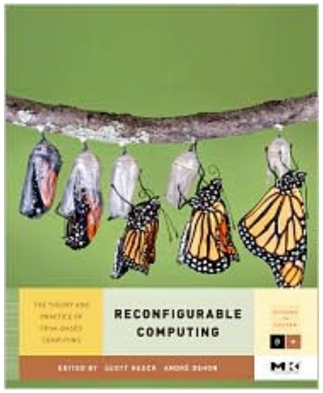
[DeHon 04]

Class organization

- Grade assignments:



- Sources: book, papers, lecture slides, manuals, and book chapters.



[editors: Hauck & DeHon]

- Class website:
  - http://scale.engin.brown.edu/classes/EN2911XF09