# Design of a RISC Microcontroller in **48** Hours

**D. Sulik** [1(2)] **M. Vasilko** [1]

[1] Microelectronics Research Group
School of Design, Engineering & Computing
Bournemouth University

[2] Department of Microelectronics
Slovak University of Technology

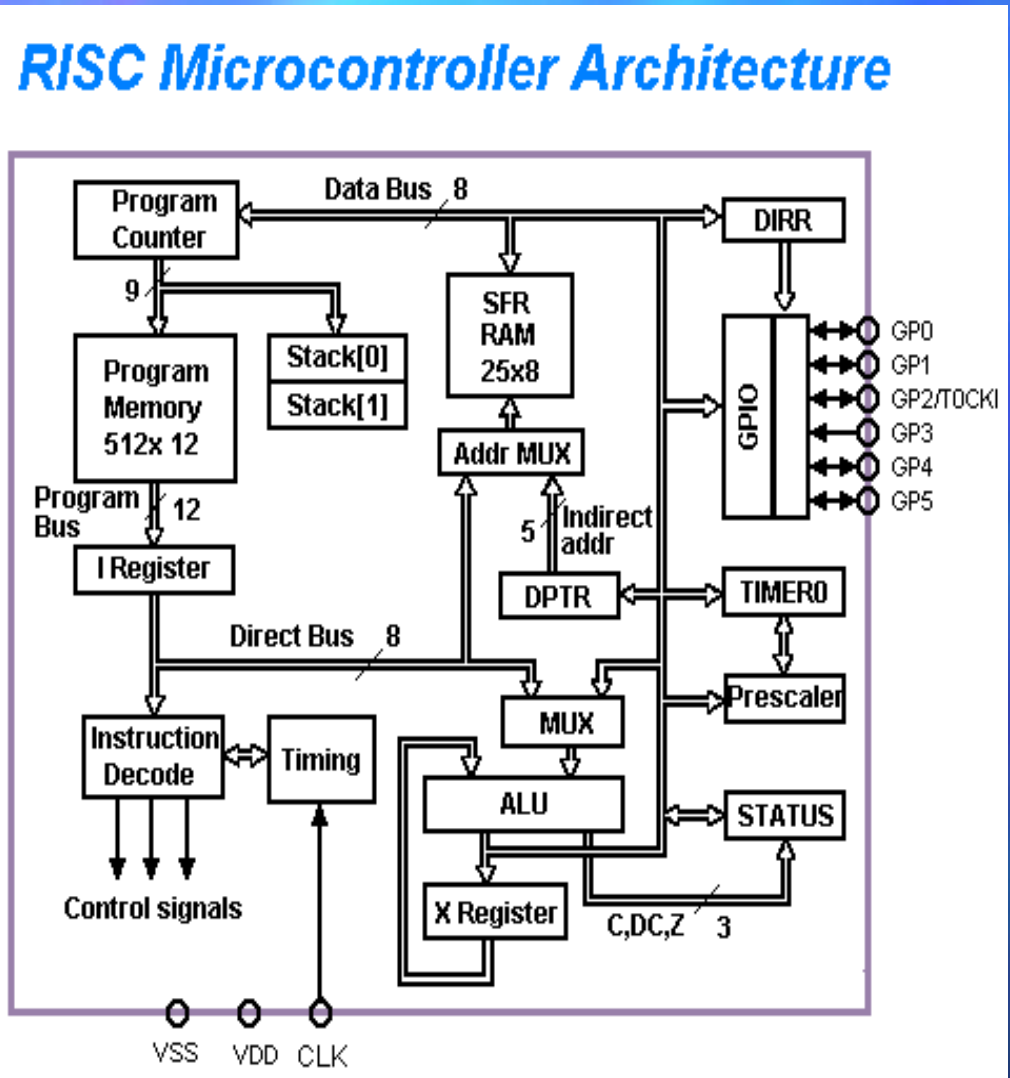
# Outline

- Introduction
  - HDLs & HW compilation
- RISC CPU Architecture
- Handel-C, what is it?
  - Handel-C and Design Flow
- RISC CPU Implementation using Handel-C
  - Code examples and Timing
  - Project  Timescale
- Conclusions

# Introduction

- Schematic Entry
- Verilog/VHDL - synthesisable
- System-level design languages
- HW compilation:
  - Handel-C, C-like language
  - fast & easy design
  - easy to learn

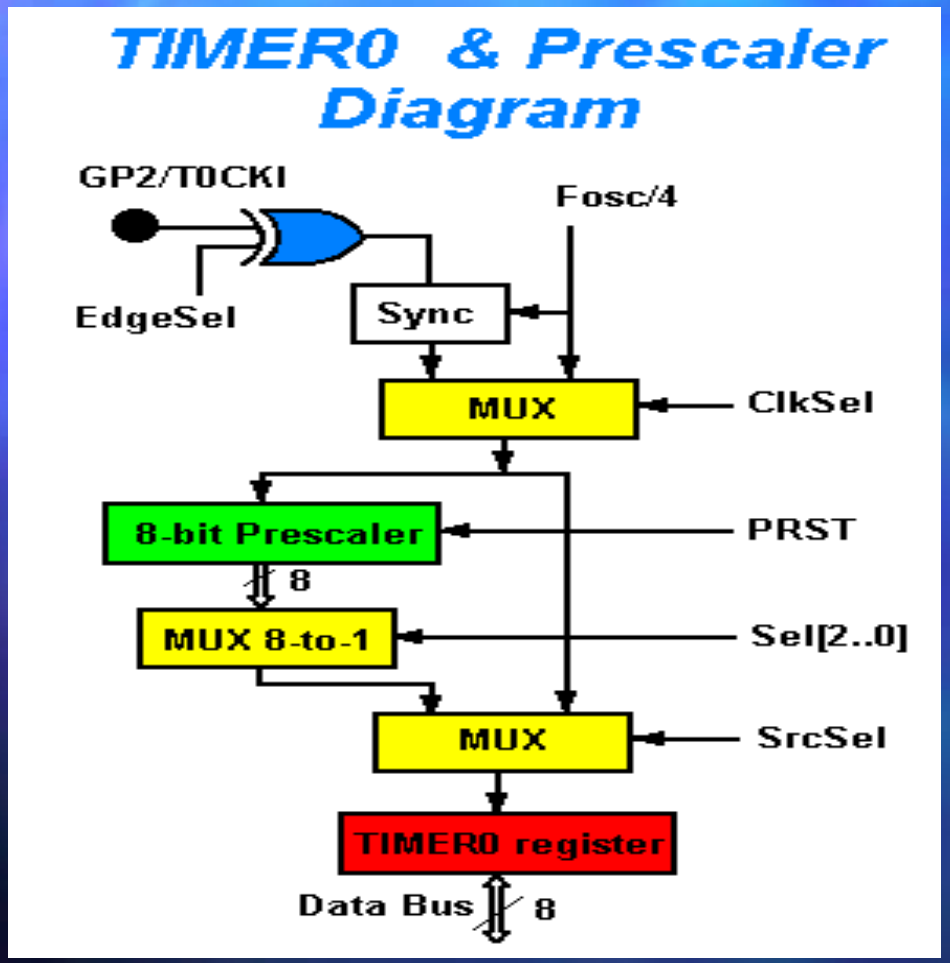
# RISC CPU architecture




- PIC* Based Architecture
- Harvard Architecture
- 512 x 12 Program Memory
- 25 Bytes Internal Data RAM
- Two level deep hardware stack
- 8 bit RTC/Event Counter (TIMER0)
- Programmable Prescaler
- 33 instructions
- Direct & indirect addressing
- SLEEP mode
- Wake-up from on pin change

* PIC is a registered trademark of Microchip Technology Inc.

# TIMER0 & PRESCALER Peripheral Module

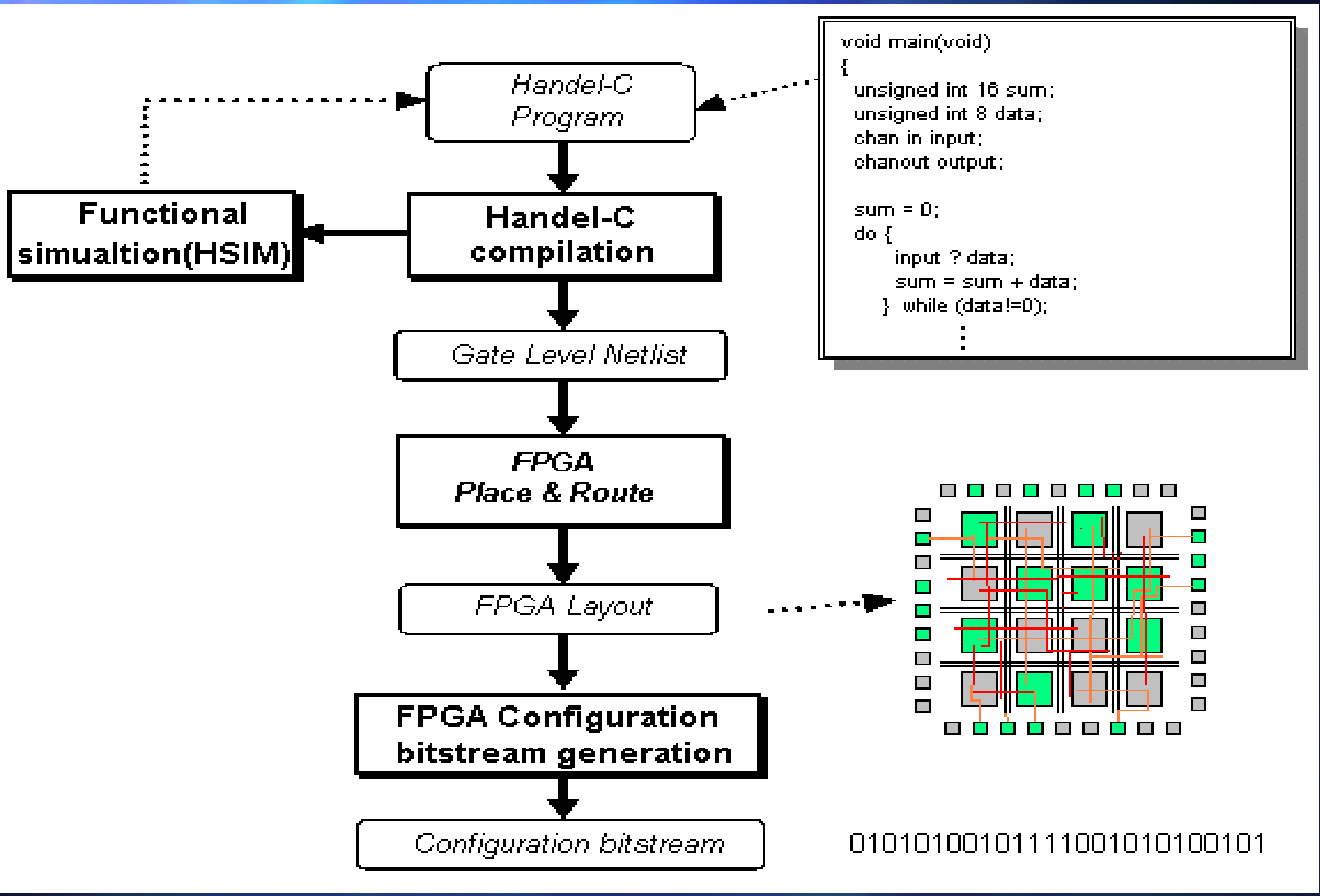



- 8-bit TIMER0 register
- 8-bit Prescaler
- Internal/External clock source
- Increment on Rise/Fall Edge of pin GP2
- Option to assign Prescaler output to TIMER0

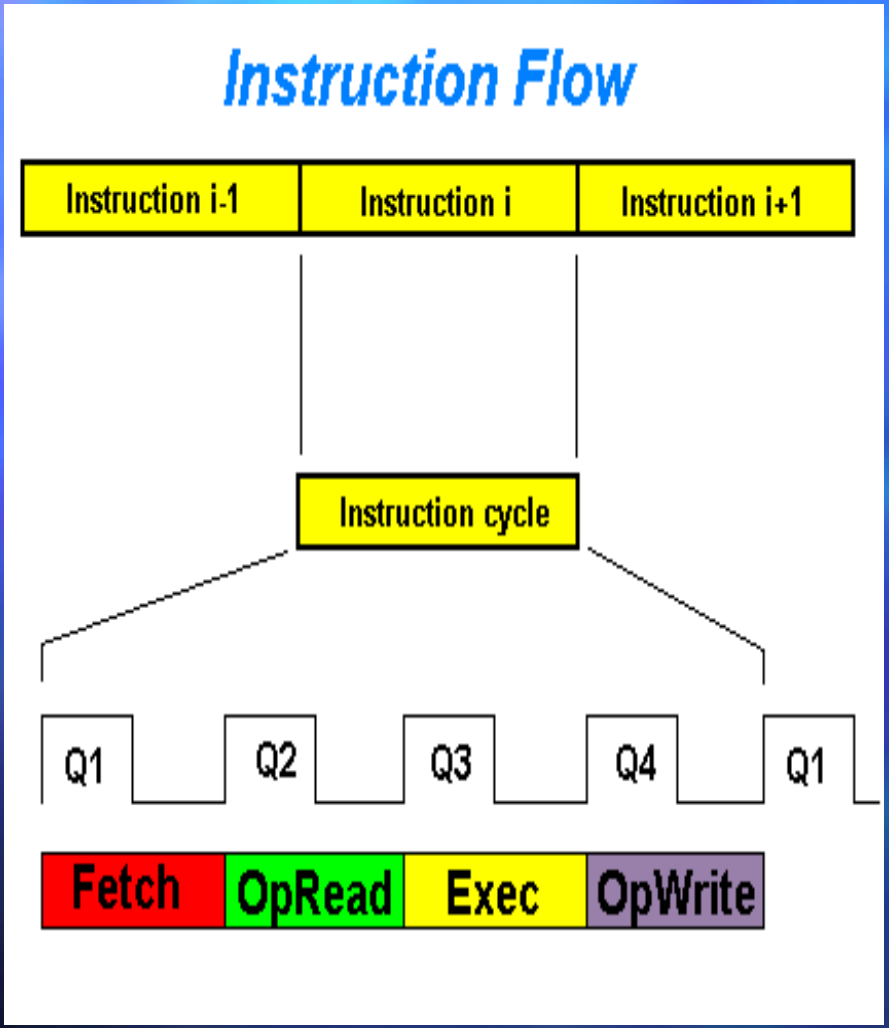
# What is Handel-C ?

- “It’s not HDL but Programming language”
- Direct C-like code compilation to HW
- Outputs to Xilinx or Altera netlist

# Handel-C Design Flow



# Handel-C implementation of CPU




```
void main(void)
par {
        {
          fetch();
          operandREAD();
          exec();
          operandWRITE();
        }
        tmr0();
      }
```

# Instruction fetch & Operand Read Phase

- Instruction fetch

```
macro expr fetch()
par {
        IR = PROGRAM[PC];
        PC = PC +1;
      } // End of par{}
```

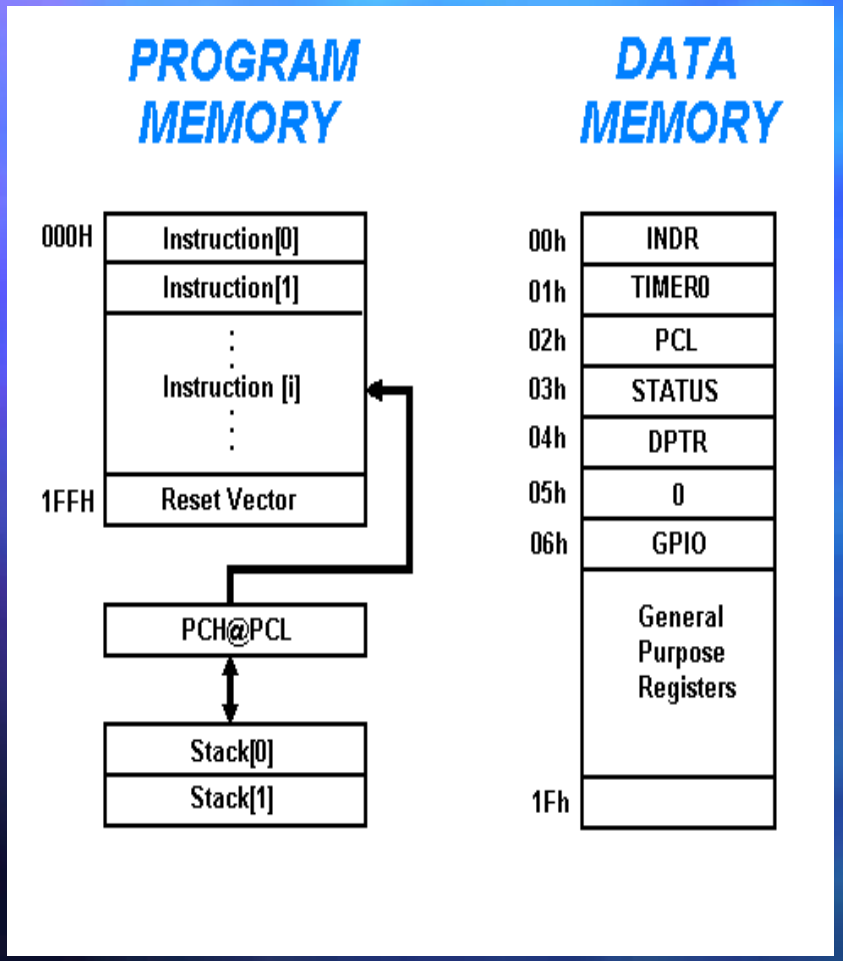
- Operand read:

```
macro expr operandREAD()
{
 switch(IR[4:0]) {
  case 0: TMP = 0;
  case 1: TMP = TMR0;
  case 2: TMP = PCL;
  ...
  case 6: TMP = GPIO;
  default: TMP=  DATARAM[IR[4:0]-7];
   } //End of switch()
}    //End of macro expr
```

# Execution phase

```
macro expr exec()
{
  switch (IR[11:5]) {
    case 0b000000://NOP
                        delay;
                        break;
    case 0b000111: //ADDXF
                       par {
                             TMP = X+TMP;
                             bCarry = carry(X,TMP);
                             bDcarry = dcarry(X,TMP);
                             bZero = zero(X,TMP);
                           }
    case ...
}
```

# Memory and I/O Implementation in Handel-C




- Program memory declaration:

```
rom unsigned int 12 PROGRAM[512]
with { offchip = 1, data = pins,  addr =
pins,   we = pins,  oe =pins,  cs = pins };
```

- Data memory declaration:

```
ram unsigned int 8 DATARAM[25]
with { data, ...};
```

- I/O declaration:

```
interface
bus_ts (int 1) IOPORT1(GPIO[1], DIRR[1]==0)
with {data = {P12}};
```

# Methodology

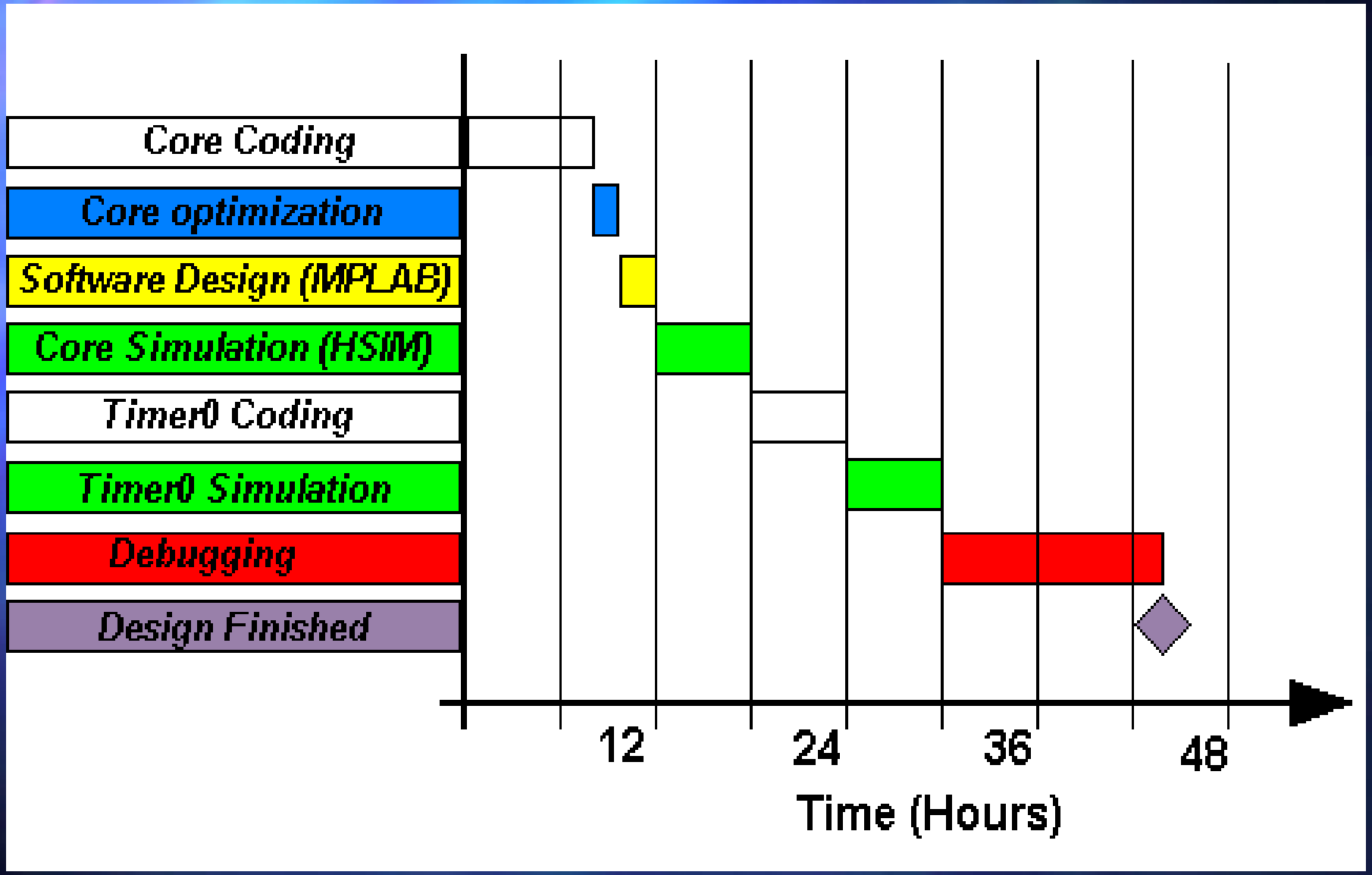
- Functional simulation
  - Handel-C Simulator (HSIM)
- FPGA implementation
  - Xilinx Alliance M2.1i
- Prototyping
  - XESS XS40 with XC4010XL
- Testing & Debugging
  - assembly language programs (MPLAB*)

* MPLAB is a registered trademark of Microchip Technology Inc.

# Implementation Statistics

- Project Resources
  - One Engineer with limited knowledge of HDLs
  - Handel-C software
  - XESS FPGA board
  - PC (Pentium 120MHz)
  - Xilinx P&R tools
  - ☺ Enthusiasm
- Statistics:
  - Design size **338 CLBs**
  - device usability **84%**
  - Clock rate **11.88 MHz**
  - Gates **884**
  - Flip-Flops **202**
  - HW compilation **5min**
  - Simulation compilation **8 secs**

# Design Time Scale



# Conclusions

- Design & Implementation & Test
- Core is soft & portable to Xilinx, Altera, ASIC technologies
- Fast coding time
- Fast compilation time