Axilica

Accelerating Silicon Design

David Mulvaney
Vassilios Chouliaras

AXILICA
Accelerating Silicon Design

Loughborough University

Axilica Limited

- Axilica is a spin out from Electronic and Electrical Engineering at Loughborough
- Company formed in September 2007 with investment from IPSO Ventures and Lachesis
- Staff
  - VP business development/project manager
  - 5 technical development team members
  - 2 consultants
  - Admin, legal and financial staff (IPSO)

Axilica history

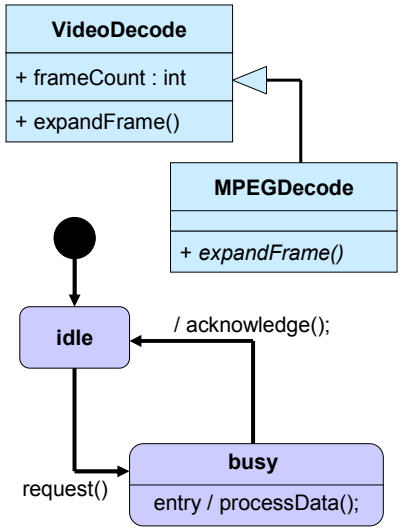
- Oct 2004 - Sept 2005 CTES scholarship
  - *original concept*
  - *broad market analysis*
- Oct 2005 - July 2006 Gatsby innovation fellowship
  - *produce prototype*
  - *develop initial plans for exploitation*
- Aug 2006 - Sept 2007 Loughborough University support
  - *produce mature prototype*
  - *patents filed*
  - *develop commercialisation route*
- Sept 2007 - Dec 2008 IPSO and Lachesis funding
  - *build company team*
  - *work with beta customers to develop product and build initial customer base*
- Jan 2009 onwards Second phase funding
  - *launch of full product*
  - *develop customer partnerships*

First product - FalconML

- FalconML is a UML design tool to accelerate chip design
- Aims to meet the market demand to continually upgrade and enhance products
- FalconML delivers a faster design flow at reduced cost and shorter time to market
- Uses UML to capture specification diagrammatically, as in software design
- Automates the laborious and time consuming manual translation of specification into hardware design code
- Currently working with beta testers in the defence, telecomms and semiconductor markets
- End of year release

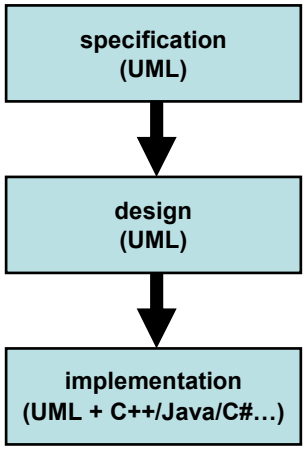
The Unified Modeling Language

- UML version 2.1.2 released Nov 2007
- De-facto standard in software engineering
- Defines 13 types of diagram, representing different aspects of a system
- Supported by many commercial modelling tools



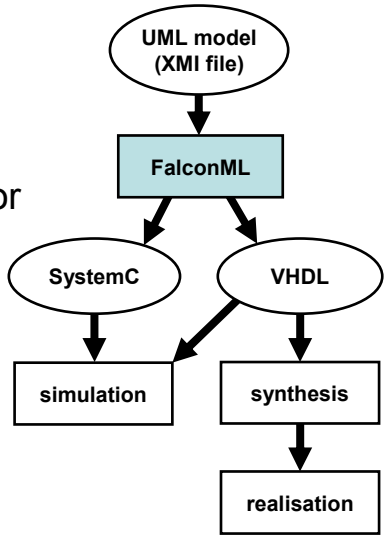
Typical software design process

- Software specification and design in UML
- Implementation in UML and C++/Java/C#...
- Tools ensure that model and code are consistent



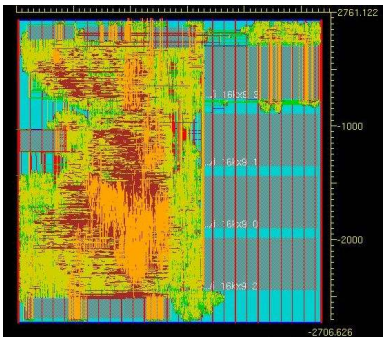
FalconML design flow

- Specification, design and implementation in UML and C++
- SystemC generated for rapid functional simulation
- VHDL generated for hardware implementation



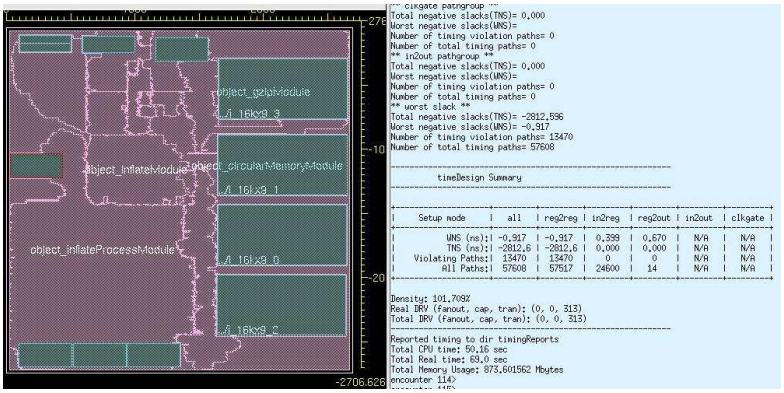
Example designs

- MD5 cryptographic checksum
- Mandelbrot/Julia fractal rendering
- GZip decompression
- JM H264 decoder



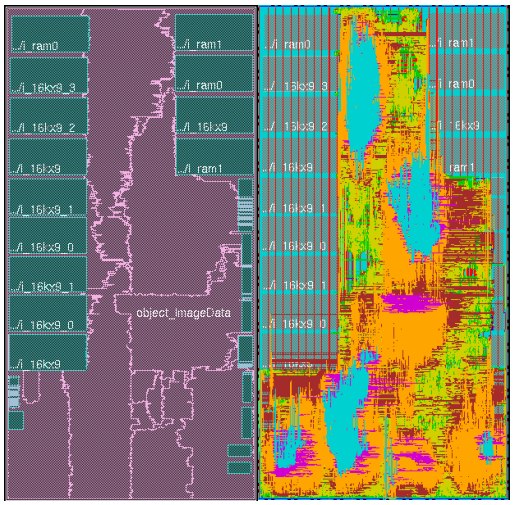
Layout for the GZip decompression unit synthesised for TSMC 0.13 µm ASIC technology

GZip deflate



WNS=-0.917
Cycle time = 5-(-0.917) = 5.917 or 169 MHz

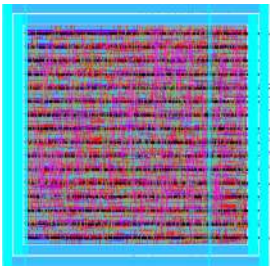
JM H264 decoder



- TSMC 0.13 µm 8M1P process
  - 110 MHz post-route
  - Gate area 5.0922 µm^2
  - Gates 2585146
  - Cells 415127
  - Area 13164081.2 µm^2
- Un-optimized P&R flow
- From UML to GDS2 in 6 hrs

AXILICA

Accelerating Silicon Design



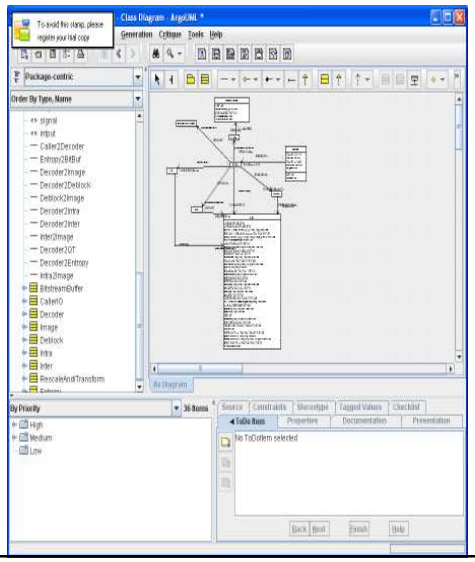
FalconML will be available for purchase in the foyer

Axilica Limited
Loughborough Innovation Centre
Epinal Way
Loughborough LE11 3EH
UK

www.axilica.com
enquires@axilica.com
+44 (0)1509-227131

JM H264 Decoder
Step 1: Design Entry

- ArgoUML 0.24
  - Classes/methods shown
- Design saved in ‘zargo’ format
- Exported as XMI for behavioural synthesis



JM H264 Decoder
Step 2: Behavioural Synthesis

- Clean command line I/F (Unix)
  - Command line I/F hidden in Windows GUI
  - Number of experimental variables available
    - Not exposed to user (yet)
- Reasonably fast compile times
- Output is
  - RTL VHDL files (one per class) and top-level design
  - Collection of SC_MODULES (one per class) and top-level
- Ready for:
  - Validation (Algorithmic)
  - Co-simulation (RTL vs Algorithm)
  - Performance modelling (Area/Timing optimization)
- On to downstream flows

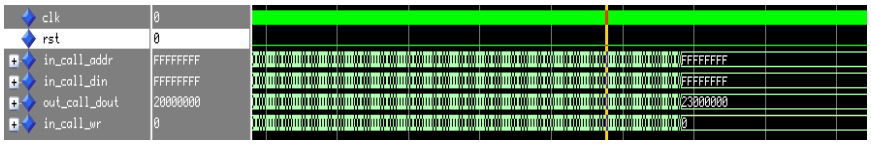
```
Linux version with no MAC locking.
Reading UML from "/mnt/hgfs/c_disk/Documents and Settings/elvc/My Documents/BUSINESS/axilica/in
tracom171207/demo/h264.xmi".
HDL will be written to ".".
--- starting processing ---
Reading /mnt/hgfs/c_disk/Documents and Settings/elvc/My Documents/BUSINESS/axilica/intracom1712
07/demo/h264.xmi
XMI read OK
Compiling model
Compiled model OK
Dumping object IDs
Synthesising RTL for model
Elaborating BitstreamBuffer
Synthesising BitstreamBuffer
Writing RTL for BitstreamBuffer
Elaborating CallerIO
Synthesising CallerIO
Writing RTL for CallerIO
Elaborating Deblock
Synthesising Deblock
Writing RTL for Deblock
Elaborating Decoder
Synthesising Decoder
Writing RTL for Decoder
Elaborating Entropy
Synthesising Entropy
Writing RTL for Entropy
Elaborating Image
Synthesising Image
Writing RTL for Image
Elaborating Inter
Synthesising Inter
Writing RTL for Inter
Elaborating Intra
Synthesising Intra
Writing RTL for Intra
```

JM H264 Decoder
Step 3: SystemC simulation

```
            SystemC 2.1.v1 --- Aug 25 2007 13:24:42
        Copyright (c) 1996-2005 by all Contributors
                    ALL RIGHTS RESERVED
Opened binary file ../../264/cat1.264 for reading
init time : 134 ns
written : 1     decoding time : 2.16006e+06 ns     running average : 2.16006e+06 ns     total t
ime : 2160062 ns
written : 2     decoding time : 1.50539e+06 ns     running average : 1.83273e+06 ns     total t
ime : 3665454 ns
written : 3     decoding time : 2.30323e+06 ns     running average : 1.98956e+06 ns     total t
ime : 5968686 ns
written : 4     decoding time : 1.67427e+06 ns     running average : 1.91074e+06 ns     total t
ime : 7642958 ns
written : 5     decoding time : 1.66765e+06 ns     running average : 1.86212e+06 ns     total t
ime : 9310604 ns
written : 6     decoding time : 1.4168e+06 ns     running average : 1.7879e+06 ns     total tim
e : 10727402 ns
written : 7     decoding time : 1.66629e+06 ns     running average : 1.77053e+06 ns     total t
ime : 12393694 ns
written : 8     decoding time : 1.2845e+06 ns     running average : 1.70977e+06 ns     total ti
me : 13678194 ns
```

- Algorithmic validation
- Performance modelling
- RTL Co-simulation

JM H264 Decoder
Step 4: VHDL RTL Verification



- I/O equivalency established across
  - Machine-generated RTL
  - SystemC model
  - Precise performance data collected
- Clean RTL for synthesis