Using Ada for Semiconductor Assembly Equipment

How ITEC uses Ada95 in semiconductor assembly equipment
ITEC within Philips

Philips

Semiconductors

Discrete

3 Assembly factories

Wafer fabs

ITEC

Let's make things better.
Assembly facilities, our customers

- Stadskanaal
- Hong Kong
- Kaohsiung
- Manila
- Calamba
- Bangkok
- Seremban

Legend:
- □ = Discretes
- ■ = ICs

Let’s make things better.
BIM line setup

BIM: Break through In Manufacturing

Endless leadframe

Adat  Phicom  Mould  Plate  Test, mark, tape
ITEC equipment

- **Adat**: Die attach machine, takes die from wafer and solders it on leadframe
- **Phicom**: Wire bond machine, bonds gold and aluminum wires from die to leadframe
- **Multiplunger**: Mould products in plastic
- **Parset**: Tests devices to meet electrical specification
- **Quad**: Machine to cut and bend leads, laser mark, test and tape products.
Started with Ada95 5 years ago

Assembly machines use RTL/2 on 68k, tester used pascal on VAXeln
Converted code to Ada95 automatically

Reasons to start with Ada 95:
- Old code can be converted to Ada95 code
- Ada95 good programing environment compared with pascal
- GNAT available, based on gcc, lots of tools available
Platform Windows NT

Why Windows NT:
• Advanced system services
• Cheap, runs on cheap hardware
• Integrates well in office environment
• Realtime properties good enough
• Advanced GUI
Selection of Ada95

+++ positive
• Conversion from rtl/2, pascal possible
• Very good from engineering point of view
• Finally good compiler (GNAT)

--- negative
• Nobody uses it (in Philips), everybody does C
• One must know C anyway
Typical Equipment control

- High end motion controllers using fast DSP’s (9kHz cycle).
- Vision function: using frame grabber to get the image, then process it on the pc
- Machine control multithreaded (typical 30-50 threads)
- Integration with Shop Floor Control system
- Support of many variants
- Response time to external events <1ms
- Setup as client/server
  - Server performs the control
  - Client interacts with user
Main architecture setup

- Client
  - user mode
  - normal priority

- Machine control
  - kernel mode
  - tcp/ip used for communication
  - Realtime priority

- NT kernel
  - WIN32 API
  - device driver
  - device driver
  - device driver
Interface with hardware & interrupts

Wrote special device driver for NT
- Allows user mode program to access physical memory
- Allows synchronization with interrupts
- User process can execute routine in kernel mode

Response times: Interrupt to thread
- average 40 microsecond
- maximum 1 millisecond
GUI clients

- Visual ITEC (Build with CLAW)
- Scope (build with CLAW)
- Curve tracer (build with gnatcom, gwindows)
Visual ITEC used as GUI interface (1)

In design mode:

- Allows to define screen layout. Select objects from application into screen.
- Can make buttons that give commands to applications. Buttons can also invoke other screens.

In execute mode:

- User makes use of screens defined.

Advantage: Application developer can focus on core of application.
Visual ITEC used as GUI interface (2)

Object types for Visual itec:
• Text objects
• Chart objects
• Life video objects
• Wafermap objects (gives view of wafer with die processed)

Visual itec characteristics:
• Inquires server periodically to get its information
• Can run anywhere because of tcp/ip connection
• Can combine information from different servers in one form
Visual ITEC used as GUI interface (3)
Scope debugging tool

Scope architecture

• In our application we use circular buffers to store events with their time. Such an object is called a probe.
• Application can put time stamps with boolean in them
• Application can put floating point value with time in them.
• To get timestamps, we use rdtsc of pentium
• Setting a timestamp is very fast << 1us
• Application has low priority server that allows client access to data
• Client displays data graphically
<table>
<thead>
<tr>
<th>Function</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMACWE 0</td>
<td></td>
</tr>
<tr>
<td>X_MOTOR_W</td>
<td></td>
</tr>
<tr>
<td>X_MOTOR_IP</td>
<td></td>
</tr>
<tr>
<td>Pmac0Com</td>
<td></td>
</tr>
<tr>
<td>X_MOTOR_P</td>
<td>4.24000E+02</td>
</tr>
<tr>
<td>X_MOTOR_E</td>
<td>4.27782E+02</td>
</tr>
<tr>
<td>X_error</td>
<td>-3.78212E+00</td>
</tr>
<tr>
<td>X_MOTOR_D</td>
<td>3.32777E-02</td>
</tr>
<tr>
<td>Y_MOTOR_W</td>
<td>+ 0.184</td>
</tr>
<tr>
<td></td>
<td>782u609n</td>
</tr>
</tbody>
</table>
Conclusions

• Ada95 is very good programming language
• Ada95 tasking features makes machine control much simpler
• Ada95 gives stable applications because of its build in protection mechanisms
• Ada95 made for proper software engineering
• Ada specs gives developer good idea of package content. When properly setup increases reuse of packages.
• In the beginning NT support was just enough. Getting better each year: stable gnat compiler, gnatcom, gwindows