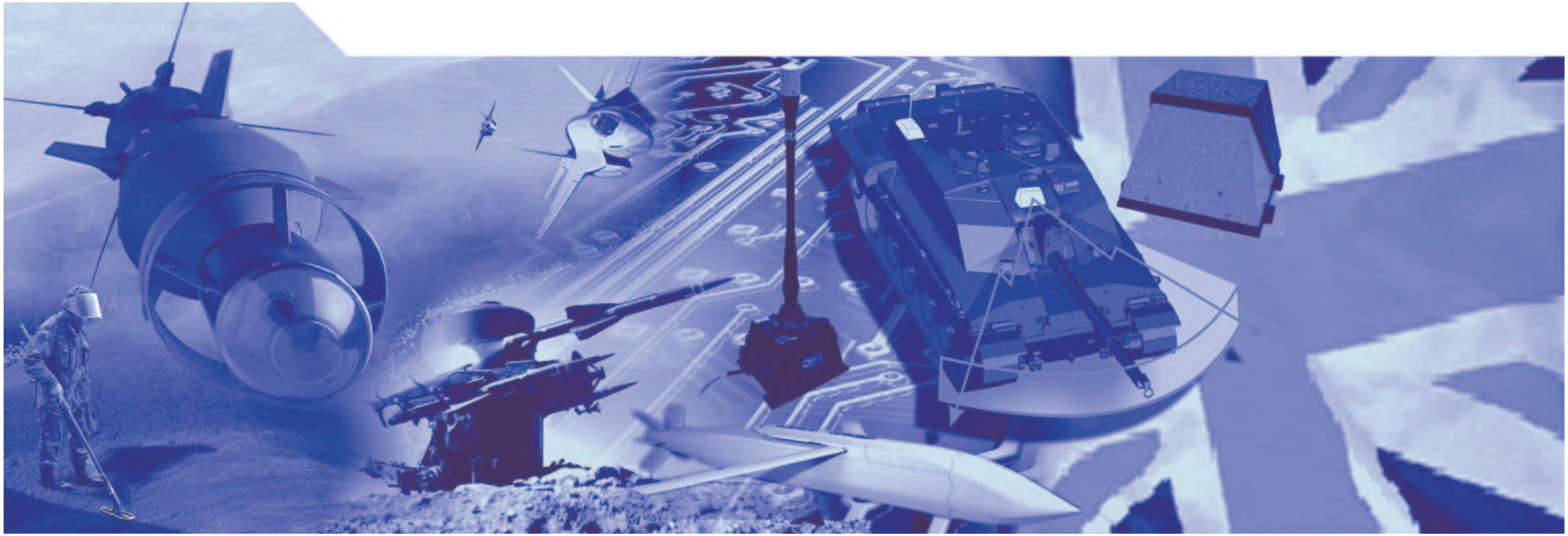
THALES



Software Development with Real-Time Workshop Embedded Coder

Nigel Holliday
Thales Missile Electronics

Contents

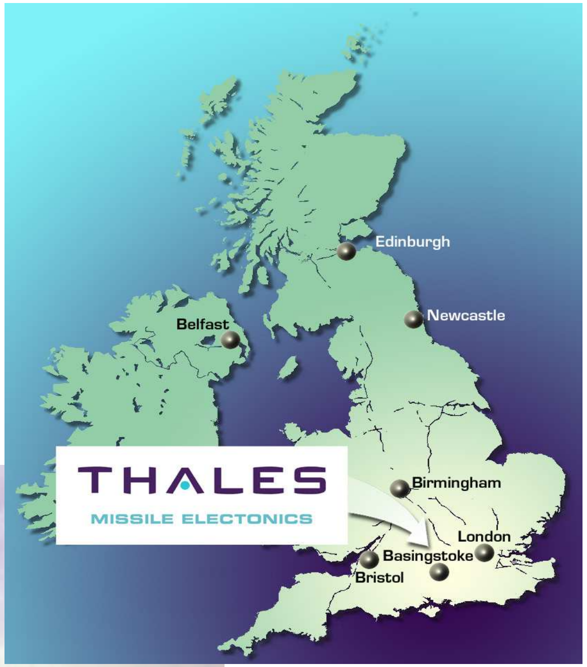
- Who are we, where are we, what do we do
- Why do we want to use Model-Based Design
- Our Approach to Model-Based Design
- Where did we use Model-Based Design
- What benefits were seen
- What difficulties did/do we experience
- Where do we want to go now
- Conclusions so far

Who are we - Thales UK Operations in Defence

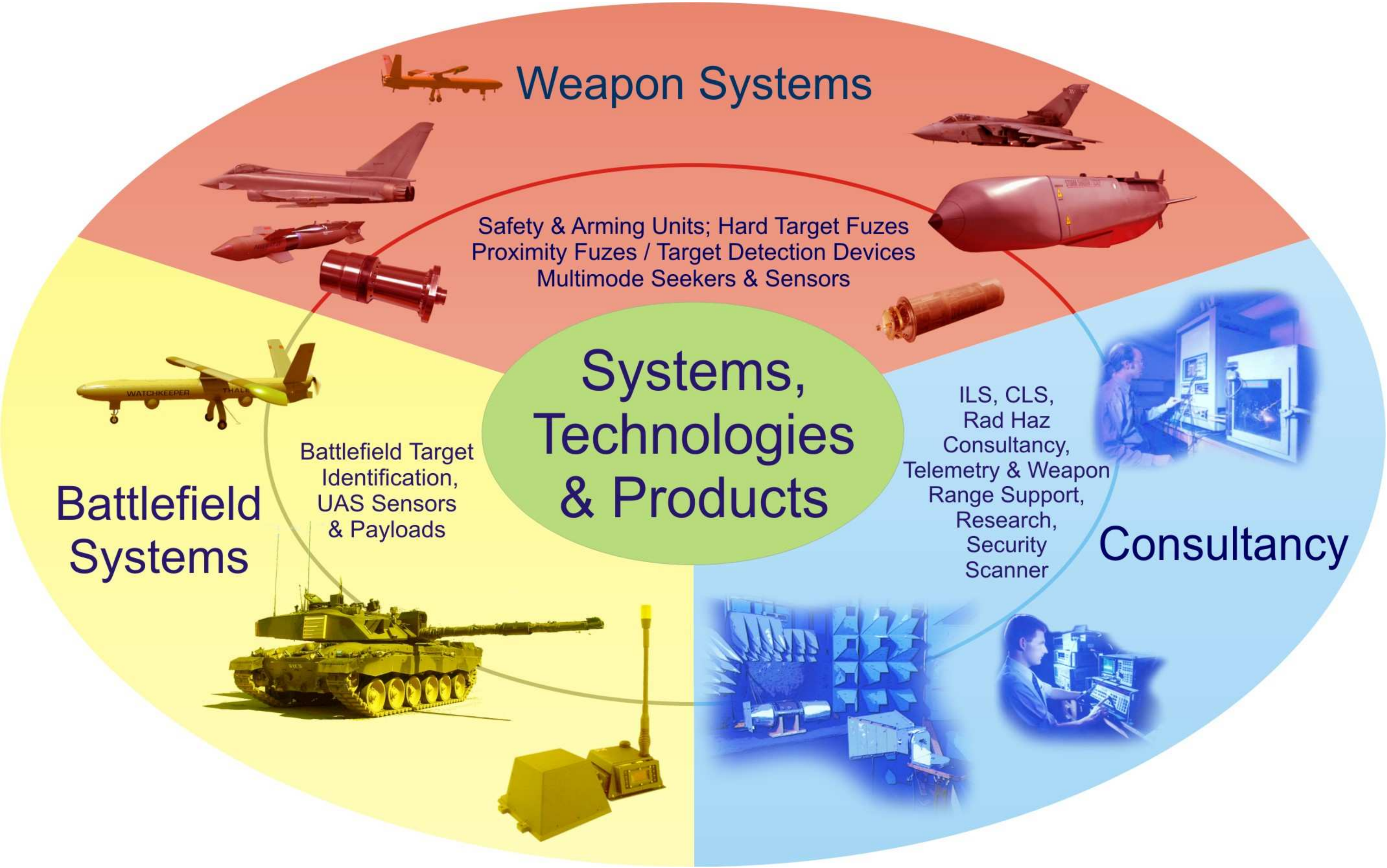
- 2nd largest defence systems contractor in the UK
- Operates at 3 levels in the UK market
  - Prime contractor
  - Sub-system integrator – where we take responsibility for integrating complete sub-systems for a platform
  - Sub-system supplier – where we will offer in competition world class technology and / or products
- Building on our core systems integration capability
- Growing CLS (Customer Logistic Support) business

Where are we – Thales Missile Electronics

- TME: Basingstoke
  - Single Integrated Site
    - On-site manufacturing
    - Laboratories
    - Environmental test facilities
    - 240 staff



What do we do



Weapon Systems

Safety & Arming Units; Hard Target Fuzes
Proximity Fuzes / Target Detection Devices
Multimode Seekers & Sensors

Systems, Technologies & Products

Battlefield Systems

Battlefield Target Identification, UAS Sensors & Payloads

ILS, CLS, Rad Haz Consultancy, Telemetry & Weapon Range Support, Research, Security Scanner

Consultancy

Why do we want to use Model-Based Design

- Save money!
  - Reduce coding effort and timescales
  - Reduce introduction of errors – reduced risk
- Reduce the need for documentation
  - Requirements - DOORS
  - Design specifications – lost in translation!!
    - The model is the design – graphical solution but well documented

Why do we want to use Model-Based Design

- Rapid prototyping
  - Early checking of software on target - timing/resources
    - Functional correctness of algorithms
    - Determine run-time and memory requirements
  - Design decisions on target hardware
    - Put on eval boards quickly to confirm following
      - 16-bit or 32-bit
      - Floating or fixed point?
      - Memory – internal/external?
      - FPGA required?

Why do we want to use Model-Based Design

- More efficient use of resources
  - Modelling engineers concentrate on creating the model and supporting real-world environments
  - Embedded engineers concentrate on processor scheduling and I/O to the rest of the physical system
    - The model plugs into the embedded software harness
  - Uptake of Model-Based Design could lead to less distinction between the two disciplines
    - Increased labour flexibility – common toolsets
    - Hybrid engineers!!
    - Broader understanding of design and implementation

Our Approach to Model-Based Design

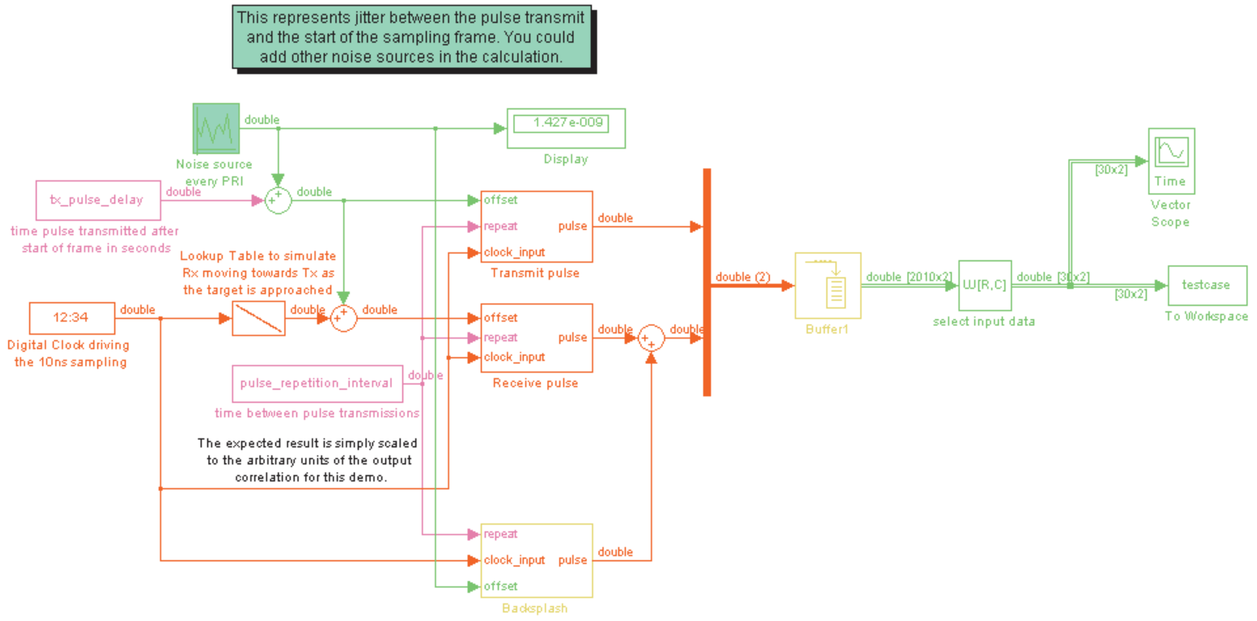
- TME approach to Model-Based Design was not to use it in the harness
  - Decision at the start of the pilot project was the model was to plug into a hand-coded scheduler/harness
  - C coding was used for all software programming of the target resources
  - Model could be taken from the Simulink “real-world” environment and C code generated
    - Some processor I/O simulation in real-world environment where required

Where did we use Model-Based Design

- Two projects used MBD
  - P1: Data processing for a single channel pulsed proximity sensor + timing algorithm
    - TME designed custom hardware for TDP
    - Software developed for 2 x dual-core 16-bit fixed-point DSPs
      - Serial and parallel I/O required with DMA
      - FPGA + analogue front-end
  - P2: Control algorithms for a gimbal assembly with mounted pulsed laser and PIR dual mode sensing
    - COTS hardware with 4 x floating-point DSPs
    - Single DSP used to run model
      - Parallel I/O
      - FPGA – gateway to rest of the system
    - Vendor board support library

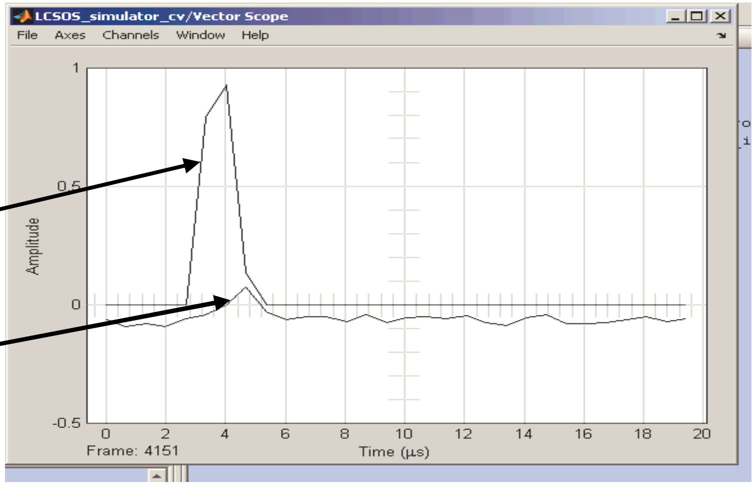
Where did we use Model-Based Design: P1

Simulator


This represents jitter between the pulse transmit and the start of the sampling frame. You could add other noise sources in the calculation.
double
Noise source every PRI
1.427e-009
Display
tx_pulse_delay
double
time pulse transmitted after start of frame in seconds
double
offset
repeat
pulse
clock_input
double
Transmit pulse
Lookup Table to simulate Rx moving towards Tx as the target is approached
12:34
double
Digital Clock driving the 10ns sampling
double
double
offset
repeat
pulse
clock_input
Receive pulse
double
double
pulse_repetition_interval
double
time between pulse transmissions
The expected result is simply scaled to the arbitrary units of the output correlation for this demo.
repeat
clock_input
pulse
offset
double
Backsplash
double (2)
Buffer1
double [2010x2]
U[R,C]
select input data
double [30x2]
[30x2]
Time Vector Scope
[30x2]
testcase
To Workspace

- Create representative simulator
- Historic Information
- Use measured results

Data saved as .mat file



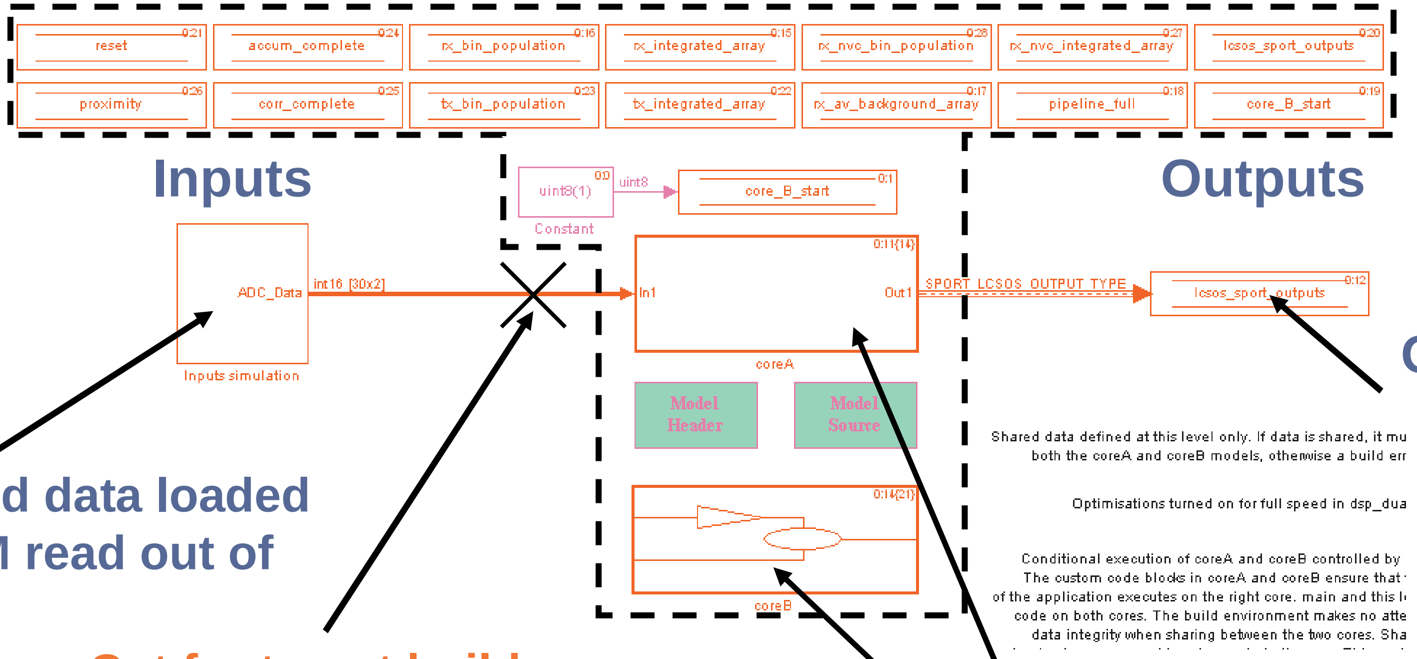
Tx pulse with noise

Rx pulse with noise & range law

THALES

Where did we use Model-Based Design: P1

Model



Shared data defined at this level only. If data is shared, it mu
both the coreA and coreB models, otherwise a build err

Optimisations turned on for full speed in dsp_dua

Conditional execution of coreA and coreB controlled by
The custom code blocks in coreA and coreB ensure that
of the application executes on the right core. main and this l
code on both cores. The build environment makes no atte
data integrity when sharing between the two cores. Sha

Inputs

Outputs

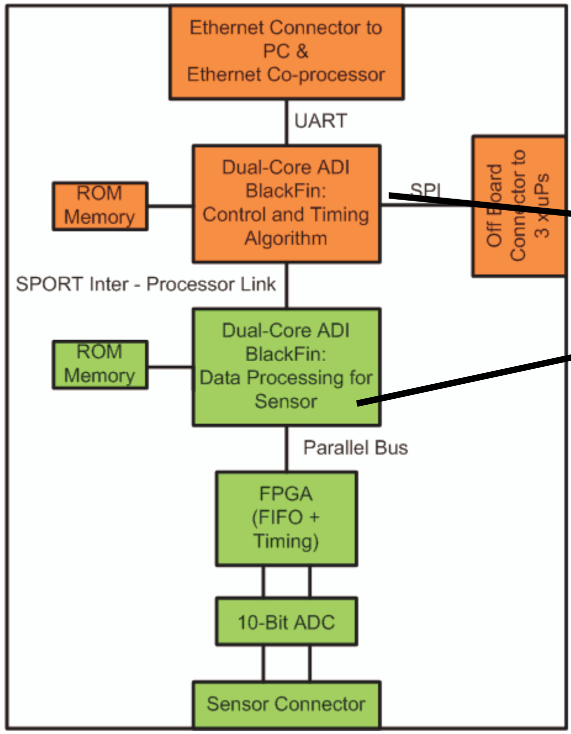
Output to harness

Simulated data loaded into RAM read out of memory

Cut for target build

Models for Dual-Core DSP

Where did we use Model-Based Design: P1



Ethernet Connector to PC & Ethernet Co-processor

UART

ROM Memory

Dual-Core ADI BlackFin: Control and Timing Algorithm

SPI

Off Board Connector to 3 x uPs

SPORT Inter - Processor Link

ROM Memory

Dual-Core ADI BlackFin: Data Processing for Sensor

Parallel Bus

FPGA (FIFO + Timing)

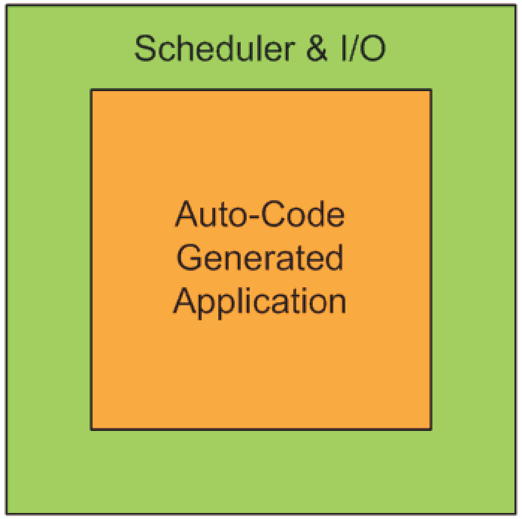
10-Bit ADC

Sensor Connector

Generic Sub-System Colour Scheme

Processor 1	🟧
Processor 2	🟩

Simple Ideology

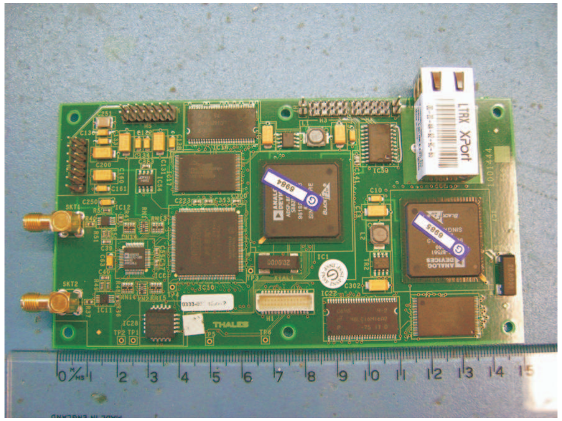


Scheduler & I/O

Auto-Code Generated Application

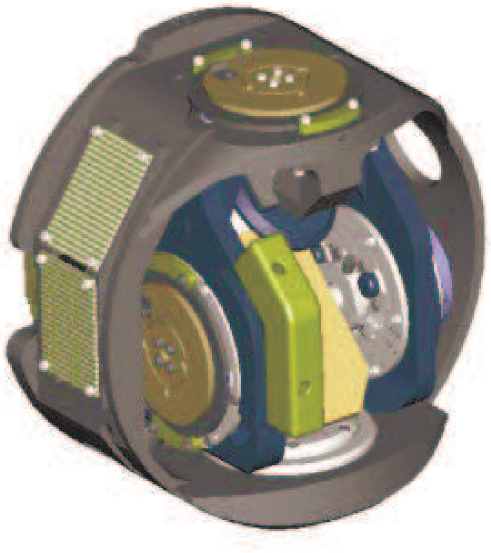
Generic scheduler with I/O for processor family

TME Custom Hardware



Where did we use Model-Based Design: P2

- Simulation
  - Real-world model in Simulink
    - Several modes required
    - Single mode simulation model optimal– time/cost v payback
  - Gimbal model developed in ProE



Where did we use Model-Based Design: P2

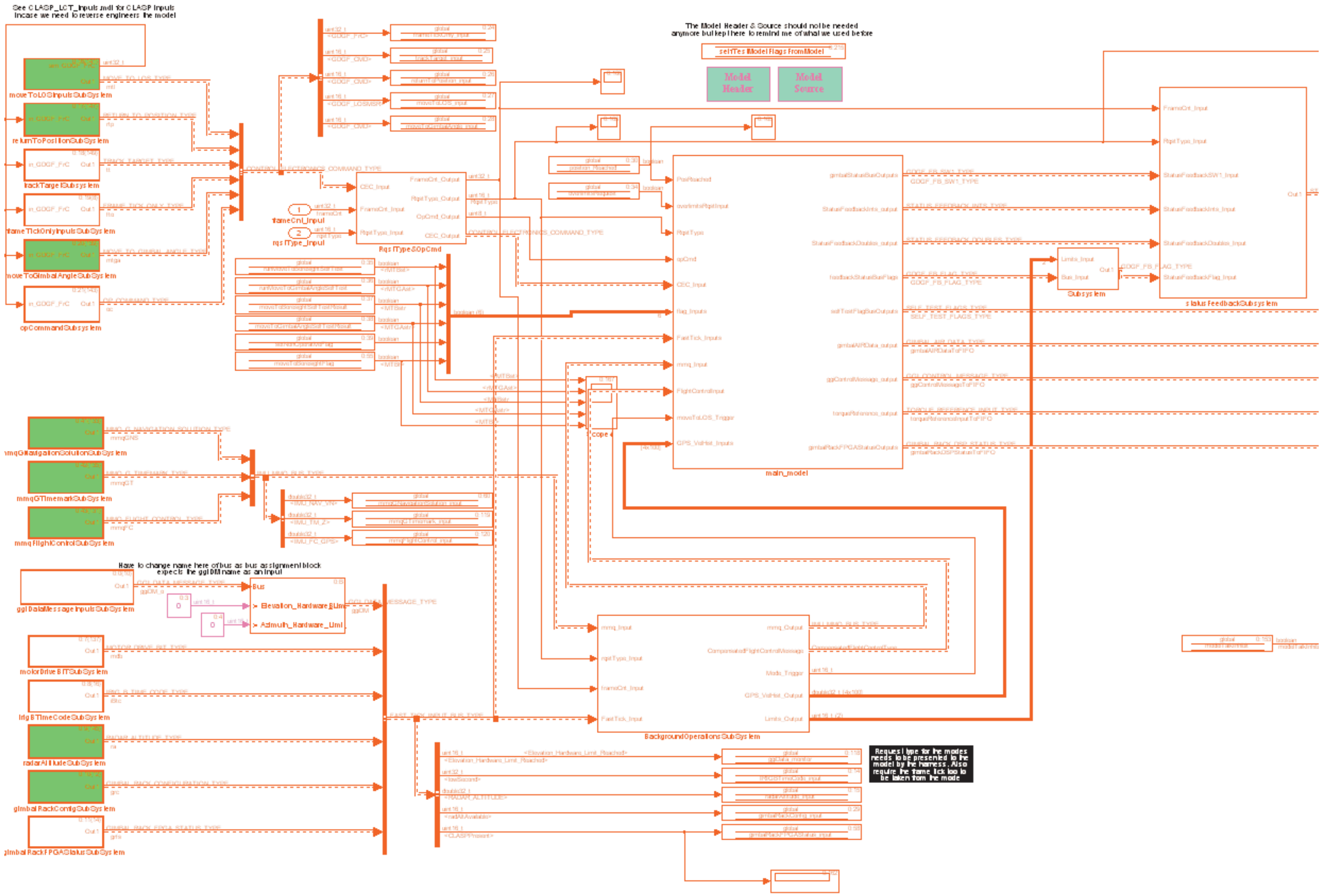
- Inputs derived from real-world model

- Model evaluated on hardware and compared against simulation for timing & correctness – it does what it says on the can

Inputs

Model

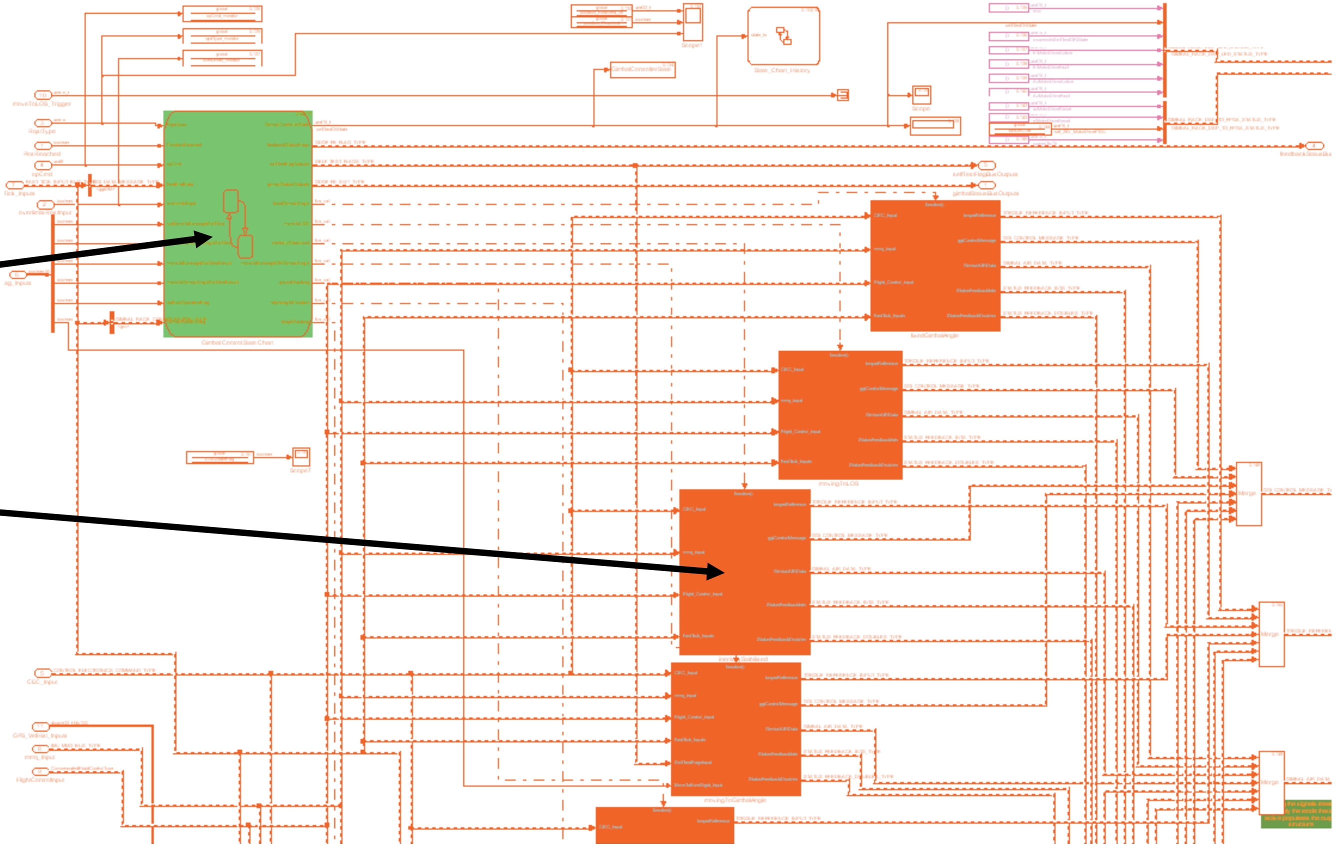
Outputs



Top Level Model

Where did we use Model-Based Design: P2

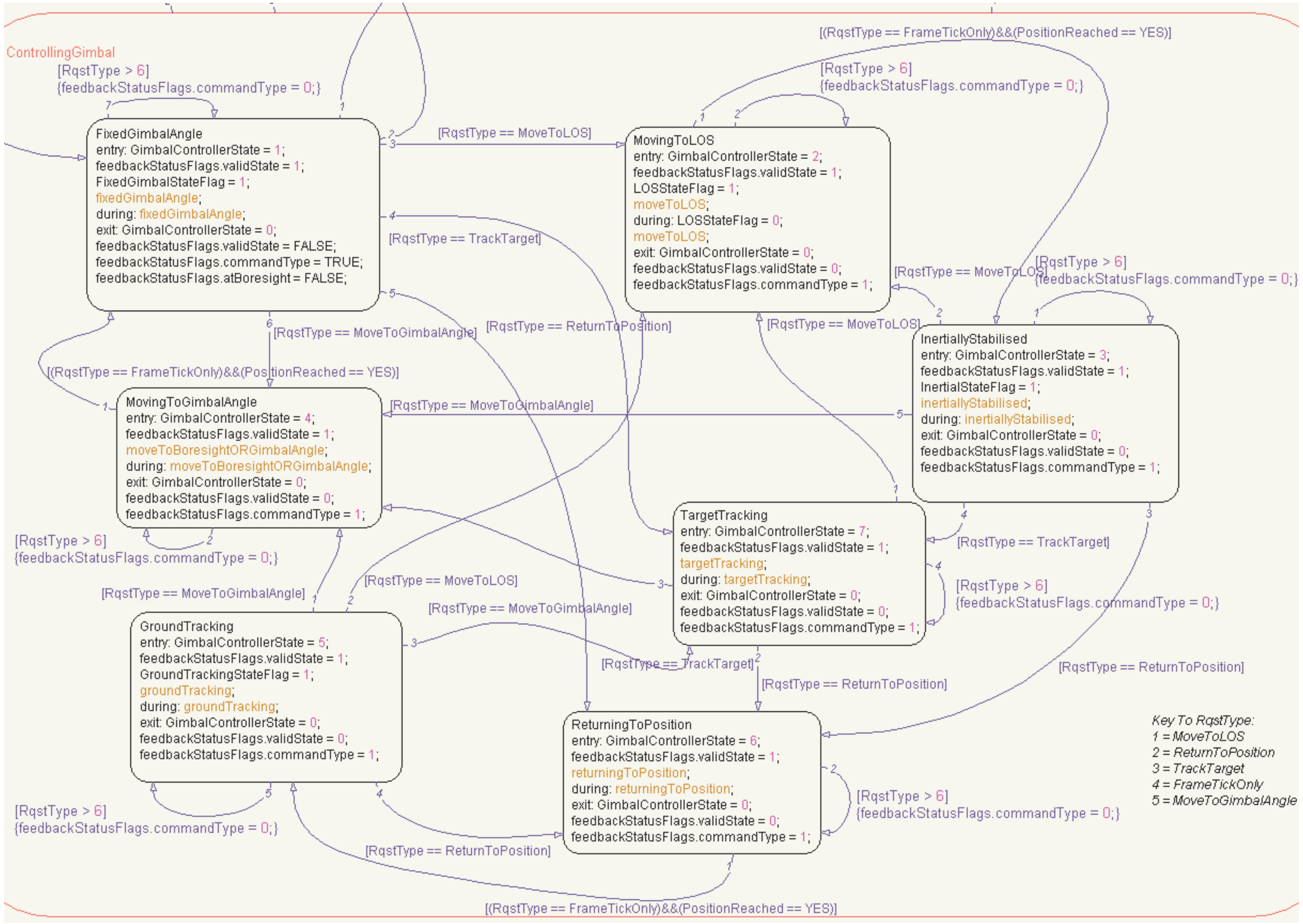
Second Level Model



- **State-machine implemented in state-flow**
- **Modes/States picked from original simulink model**

Where did we use Model-Based Design: P2

Gimbal State Controller



ControllingGimbal

```
FixedGimbalAngle
entry: GimbalControllerState = 1;
feedbackStatusFlags.validState = 1;
FixedGimbalStateFlag = 1;
fixedGimbalAngle;
during: fixedGimbalAngle;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = FALSE;
feedbackStatusFlags.commandType = TRUE;
feedbackStatusFlags.atBoresight = FALSE;
```

```
MovingToLOS
entry: GimbalControllerState = 2;
feedbackStatusFlags.validState = 1;
LOSStateFlag = 1;
moveToLOS;
during: LOSStateFlag = 0;
moveToLOS;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

```
InertiallyStabilised
entry: GimbalControllerState = 3;
feedbackStatusFlags.validState = 1;
InertialStateFlag = 1;
inertiallyStabilised;
during: inertiallyStabilised;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

```
MovingToGimbalAngle
entry: GimbalControllerState = 4;
feedbackStatusFlags.validState = 1;
moveToBoresightORGimbalAngle;
during: moveToBoresightORGimbalAngle;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

```
TargetTracking
entry: GimbalControllerState = 7;
feedbackStatusFlags.validState = 1;
targetTracking;
during: targetTracking;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

```
GroundTracking
entry: GimbalControllerState = 5;
feedbackStatusFlags.validState = 1;
GroundTrackingStateFlag = 1;
groundTracking;
during: groundTracking;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

```
ReturningToPosition
entry: GimbalControllerState = 6;
feedbackStatusFlags.validState = 1;
returningToPosition;
during: returningToPosition;
exit: GimbalControllerState = 0;
feedbackStatusFlags.validState = 0;
feedbackStatusFlags.commandType = 1;
```

[(RqstType == FrameTickOnly)&&(PositionReached == YES)]
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[RqstType == MoveToLOS]
[RqstType == TrackTarget]
[RqstType == MoveToGimbalAngle]
[RqstType == ReturnToPosition]
[RqstType == MoveToLOS]
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[(RqstType == FrameTickOnly)&&(PositionReached == YES)]
[RqstType == MoveToGimbalAngle]
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[RqstType == MoveToGimbalAngle]
[RqstType == MoveToLOS]
[RqstType == MoveToGimbalAngle]
[RqstType == TrackTarget]
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[RqstType == TrackTarget]
[RqstType == ReturnToPosition]
[RqstType == ReturnToPosition]
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[RqstType > 6]
{feedbackStatusFlags.commandType = 0;}
[RqstType == ReturnToPosition]
[(RqstType == FrameTickOnly)&&(PositionReached == YES)]
1
2
3
4
5
6

Key To RqstType:
1 = MoveToLOS
2 = ReturnToPosition
3 = TrackTarget
4 = FrameTickOnly
5 = MoveToGimbalAngle

What benefits were seen

- Re-use of simulation data
  - Same stimuli used for model verification on hardware
    - Easy/fast capture of test stimuli for model from real-world model
    - Cross referencing simulation and hardware model versions
- Rapid prototyping possible
  - Extensive use of low cost microprocessor evaluation boards prior to making hardware decisions
  - Evaluate model and hardware it is to run on
  - Timing analysis/profiling – can the model run fast enough on hardware
  - Optimise parts of model if necessary

What benefits were seen

- Reduced specification writing
  - No need for lengthy detailed design specs
    - Well documented model with graphical flow can yield almost as much detail as a written specification – can do this in the model
    - Well organised model with several tiers can clearly show model hierarchy (with adequate labelling)
    - Software interface documentation still required
- Rapid response to change/additions to requirements
  - New model sections rapidly integrated and tested on hardware
    - Maximise use of existing architecture – greater visibility with graphical model

What benefits were seen

- Powerful linkage between model and software run on the hardware established
  - During integration can return easily to model for debug
    - Simulink display facilities allow easy visibility for rapid debug
    - Still use microprocessor development environment
      - Breakpoints
      - Memory/register contents
      - Execution time
    - Can aid debug of third party sub-systems
- No perceivable increase in development time during the learning curve period
  - Scheduler required significant development time
  - This needs to be done anyway

What benefits were seen

- Ability to review model with third party
  - TME program management team
  - Customer
  - Other team project members
  - Internal review processes

What difficulties did/do we experience

- Where to start!!!
  - No prior experience of Simulink or Stateflow
    - Mathworks training courses only in 2005
  - How to architecture the model for simulation
  - Limited experience of house keeping activities for code generation from a Simulink model
    - Template Make Files
    - Low level understanding of compiler options
    - Code and data placement in memory

What difficulties did/do we experience

- Pressures to deliver on a live project
  - Learning curve to go up
- Debugging the model
  - Setting breakpoints in the model
  - Is it Simulink or the target environment
  - Program flow through the model
    - Graphical interpretation of execution order
    - Program control sometime difficult to understand
    - In-built debugger hard to drive – lack of training/experience?

What difficulties did/do we experience

- How to configure a model for multiple developers
  - TME uses Sourcesafe for software
  - How do we handle multiple developers on a single model for configuration and integration – even for desktop development
    - More acute for embedded applications

Where do we go now

- Demonstrate significant reductions in timescales for model based development
  - Acceptance by program managers and company hierarchy only if visible savings
- Define a company process for model based design involving code generation
  - Record current knowledge so not lost!
  - Iterative/learning process
- Use on more projects
  - Increase expertise in model based design across the company product range and staff – where applicable

Where do we go now

- MISRA compliant hand/model generated code
  - Future products expected to require safety related software
  - Increase documentation within the models
- Make use of linkage with DOORS
  - For bigger programs
  - Simplify requirements and compliance management
- Make more use of in-built Simulink reporting tools to better describe model – the model is the specification

Conclusions so far

- No perceived increase in development time/cost in early programs
  - Savings masked by other activities that are also on the learning curve – e.g. new processor
- If it happens in the model it will happen on the target
- Re-use of simulation data allows early evaluation of algorithms/models on target resources
- Model-Based Design very flexible and responsive to change (for example dual vs. single core)

Conclusions so far

- Still work to do to define a process
  - Iterative activity to get to a process that works
  - Flexible process to cater for desktop and embedded applications
- MathWorks pilot support throughout - Excellent!

- Similar pilot study evaluating Model-Based Design carried out at a Thales sister company in Belfast
  - Automatically generated fixed point code ran 30% faster than the hand written fixed point code