Data Flow Modeling of Radio Applications

Orlando Moreira
Principal DSP Systems Engineer
orlando.moreira@stericsson.com
Multi-radio devices

3G Base station
HSPA

WLAN Base station

Multi-Radio Smartphones

2G Base station
GSM

GPS Satellite
Application: Radio PHY

radios are real-time applications:

- modems must meet deadlines defined by the standard
- throughput, latency requirements

![Diagram](image)

- 8 µs
- 4 µs
- Variable size N x 4 µs
- 16 µs

- Sync
- Head
- Payload
- SIFS

Deadline for sending ACK
Radios are streaming applications:

- Receiving/sending virtually infinite data sequences
- Iterative schedules with overlapped execution
- Inter-iteration dependencies
Hardware: Heterogeneous Multiprocessor

- **ARM**
  - Mem
  - A
  - NI

- **EVP**
  - Mem
  - A
  - NI

- **CoDec**
  - Mem
  - A
  - NI

- **On-chip interconnect**
  - NI
  - Source
  - Sink

- **RF**
  - A/D
  - IT
  - OT
  - MAC
Multi Radio Vision: The Radio Computer

- multiple radios run simultaneously on multiprocessor
- sharing cores, memory and communication
- radios developed and installed independently
- each radio meets hard deadlines

Radio Computer Operating System

Radio 1  Radio 2  Radio 3

Radio Computer Operating System

RF  A/D  IT  OT  MAC
SDR Vision: Software Architecture

- provide power-efficient, real-time schedules
- support wide variety of radio combinations and transitions
- allow post-design updates and add-ons
- simplify the radio design process

What if radios could share hardware?

The radio computer vision: a radio is software running on the radio computer; user installs and runs radios just like apps on a PC; multiple radios run simultaneously in shared hardware; radios are developed in isolation, delivered independently.
Refining the requirements

real-time guarantees
• automated analysis (no manually-derived models)
• no scheduling anomalies, no deadlocks, no buffer overruns

ease of programming/testing/debugging:
• correct-by-construction concurrent behavior
• code generation for communication, synchronization

efficiency:
• static scheduling whenever possible
• static determination of buffer sizes
• distributed runtime synchronization (to avoid bottlenecks)
Avoiding unpredictable behavior

**Platform:**
- Resource contention
- Unbound time to service from resource…
- … or large difference between average and worst case provision
- Results in: starvation, scheduling anomalies, over-allocation,…

**Solution:** Budgeted arbitration
- schedulers with service guarantees in all share points
Avoiding unpredictable behavior

Application:

- Non-deterministic functional behavior
- Dynamism (data-dependent behavior)
- Results in: deadlocks, halting problem, unbounded memory, ...

Solution: Restrict programming model

- Domain-specific: find right trade-off expressivity vs analyzability
For Real-time Analysis: a composable model for analysis of temporal behavior

For Software Engineering: a concurrent Component Model with strong formal properties (as opposed to UML)
**Data Flow**

**Actors:** computing stations with well-defined data-driven activation rules

**Arcs:** FIFO channels

**Tokens:** Initial data items in Arcs – imply inter-iteration dependencies (static DF)
• Longest cycle bounds maximum rate.
• Execution in bounded buffer space.
• There is always a static periodic schedule that achieves maximum rate.
• Self timed execution upper bounded by static periodic schedule.
• Monotonic, Linear in timing: No scheduling anomalies.
Data flow formalism: function + mapping + analysis

data flow unifies concurrent application specification & timing analysis of mapping

Programming Model: Specifying Functionality per RAT

Analysis Model: Modeling Mapping Decisions

Modeling of buffer sizes, static ordering, communication overhead, dynamic schedulers.
Scheduling Policy – intra vs inter graph

**intra-graph**
- dependencies known
- dependencies are static or quasi-static
- related rates of execution between tasks
- shared temporal requirements

**inter-graph**
- no dependencies
- independent start, stop
- independent rates
- independent timing requirements
- contention for resources
Scheduling Policy – intra vs inter graph

intra-graph
inter-processor synchronization: self-timed & data-driven
intra-processor: quasi-static order
- determined at compile time
- no scheduler overhead

inter-graph
per processor: budget scheduler
- guarantees per reservation
- isolates graph from interference

global resource manager
- reservation of resources,
  processor binding at graph startup
Software Architecture for SDR

Compile-Time (Budgeting)
For each graph

Run-Time (Admission Control)
For each graph start request

Clustering, Static ordering per cluster, Buffer sizes, Run-time scheduler settings per cluster

Admission control, actor to processor binding, load tasks, configure run-time schedulers
No disparity between analysis model and implementation
Dynamic Scheduler Modeling: TDM

Latency rate model [Wiggers2007]:
approximation for any starvation-free scheduler
accuracy depends on the scheduler

\[ t_{AL} = P - S(A) \]
\[ t_{AR} = \frac{P \cdot T(A)}{S(A)} \]

P: Period of the TDM scheduler
S(A): Slice allocated to A
T(A): Worst-case Execution time of A
Latency-rate server model can be used for any starvation-free/budget schedulers. It can for some cases be rather pessimistic.
Data flow Modeling: Problem with the LR-Model

Fig: The LR-model over-estimates the worst-case temporal behavior of TDM arbitration by a factor of \((P/S)\)

But do not fear. A model with precise worst-case is on the way!
Modeling TDM combined with Static Order

We can compose a data flow analysis model for a cluster of statically-ordered actors that share a slice on a TDM scheduler:

Latency component does not affect local (intra-cluster) communication.
LTE PHY – Channel Estimation

- Graph for 4 TX - 1 RX antennas
- 4 Coarse ChEst paths
- CSDF due to position of reference symbols in subframe
  - 1st, 2nd and 5th OFDM symbols
WLAN Packet structure and processing

Can Static Data flow handle this?

Deadline for sending ACK

Sync  Hea  Payload  SIFS

8 µs  4 µs  Variable size N x 4 µs  16 µs
WLAN Packet structure and processing

Sync  Hea  Payload  SIFS

Deadline for sending ACK

8 µs  4 µs  Variable size N x 4 µs  16 µs

Friday, January 27, 2012
WLAN Packet structure and processing

We can manually design a worst case model for analysis
Doesn’t work for specification, compilation, or code generation.
It is difficult, time-consuming, error prone…
…And how do we guarantee that the model is correct?
The right flavor of data flow: Expressivity
SRDF:
• deadlock free
• self-timed execution is bounded by static-periodic schedule with max rate
• static periodic schedule can be built from linear constraints
• linear/convex programming!

CSDF converts to SRDF
MRDF converts to SRDF

DDF and BDF are Turing complete, impossible to check even for deadlock freedom in the general case.
DF model for Radio: Mode-Controlled Data-flow

- allows (limited) data-dependent behavior.
- properties somewhat similar to scenario-aware data flow (TUE)
- explicit control
- It is a restriction of integer data flow [Buck]

DVB-T Receiver
3 Modes:
Sync, Decode, Drop
Our Computation model: Mode-Controlled Data-flow

... extending data flow to meet the requirements traditional data flow analysis cannot handle: latency requirements, data dependent behavior, but this is required by radio applications. eg: WLAN packet and timing...
**Our Computation model: Mode-Controlled Data-flow**

**analysis:** monotonic, strict, periodic bound per mode exists. bound self timed execution per mode, compute mode transition overhead normally limited to specific mode sequences of interest.

**scheduling:** quasi-static ordering of actors possible, bounded buffers exist
Quasi-static ordering (extension for MCDF)

- order of actors inside cluster as static as possible
- only run-time decision is mode switching
- mode synchronization among clusters handled by FIFOs
- broadcast of mode control tokens
LTE PHY Mode-Controlled Data Flow (simplified)

- modal behavior combined with cyclo-static behavior
- CHEST estimates for 1\textsuperscript{st} sample after processing 6\textsuperscript{th}
- analysis can handle it, but programming starts becoming difficult
- and what about distributed control?
- still needs more syntactic sugar…
Demonstrator (2009)

Collaboration ST-Ericsson, Nokia, NXP.
All run-time components implemented, including:
• Predictable local schedulers;
• Fifo-based communication, self-timed execution
• Resource manager, w/ runtime task and memory mapping

Best Paper Award SDR Forum
Software Architecture - Run-time of Demonstrator

- Global RM, Config Manager
- PC
- Comm Stubs (Host, RF)
- BB RM
- SoD NM
- RTOS
- FARM

Start/Stop Operational State
Status

Baseband

Fifo Comm

Real-Time

- Radio functions, MC L1
- Start/Stop Tasks, Connect Tasks
- SoD streaming kernel
- JARM

Fifo Comm

- Radio processing functions
- SoD streaming kernel
- EVP

Interconnect

[Image of the diagram with various components and interconnections]
Messages

- data flow: real-time analysis model for concurrent streaming
- data flow: concurrent programming model
- budget scheduling: independent behavior (also analysis)
- automatic generation of analysis model from implementation
- right flavor of data flow for an application is domain-specific.
THANK YOU