H.264 Decoder Algorithm Specification and Simulation in Simulink and PeaCE

School of Electrical Engineering and Computer Science
Seoul National University
Seongnam Kwon, Hyunuk Jung, Soonhoi Ha
{ksn, jung, sha}@iris.snu.ac.kr
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
Traditional Design Flow

**Steps**
- Modeling: diagram (ex. flow chart)
- Simulation & Verification: High level language (ex. C)
- Implementation: Target language (ex. C, assembly, VHDL…)

**Cons.**
- A system must be rewritten in many languages
- Translation may be inaccurate or make translation error

**The status of current system design**
- High system complexity
- High time-to-market pressure
Model-based Approach

Model-based approach

- Can help to develop embedded system to cope with ever-increasing system complexity under high time-to-market pressure
- System algorithm is specified with a block diagram or a composition of function blocks
- In most model-based tools, simulation is automatically performed with this system modeling
- Block is a unit of HW/SW partitioning, and can be reused in another system

Modeling & simulation with block diagram

Partitioning

Block reuse
Model-based Approach: Simulink

Simulink

- An extension to MATLAB® that allows developers to rapidly build computer models of dynamic systems
- System modeling: block diagram

Simulink design flow

- Automate translation from specification to simulation and to implementation
Model-based Approach: PeaCE

- **PeaCE**
  - Is a codesign environment for rapid development of heterogeneous digital systems
  - Is developed by CAPLab, SNU.

- **PeaCE design flow**
  - System simulation as well as implementation is performed with the **automatically generated codes**
    - Pros. Simulation speed, debugging capability, and code reuse
Contributions

» We present our experience of H.264 decoder algorithm specification and simulation with Simulink and PeaCE.

» We focus on three points
  ■ Algorithm modeling of multirate system
  ■ Block definition and debugging
  ■ Simulation performance
H.264 Decoder Algorithm

- H.264 is a video coding standard made by the Video Coding Experts Group (VCEG) of the ITU-T
- High bit rate saving and good video quality compared to H.263+
- High algorithm complexity
- Key features
  - Enhanced motion compensation
  - Small blocks (4x4) for transform coding
  - Improved in-loop deblocking filter
  - Enhanced entropy coding
The Block Diagram of H.264 Decoder

How can multirate be modeled?
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
Two kinds of models

- Event-driven model
  - Block is executable when event(or input) arrives to the block

- Time-driven model
  - Block is executable when time reaches block’s execution timing
  - To model multirate property, this is favorable
Use clocks, counter variables, and some buffers to model multirate property

- To know when to update the buffer and when to execute the block, counter variable is needed
- All ports of block (clocks, counter variables and buffers) should be controlled separately

```c
void block() {
    static int counter = 0;
    counter++;
    inputBuffer[counter] = read_data;
    if (counter==3) {
        counter=0;
        run(); //main code
        write_to_outputPort;
    }
}
```
Top model of H.264 decoder in Simulink

- Read Slice
- Decode MB
- Deblock & Write
Block Definition in Simulink

S-function

- a computer language description of a Simulink block
- supports C, C++, Ada, and Fortran for modeling blocks

How to use S-function?

1. Use S-function builder ➔ too restricted
2. Writing S-function block manually

Difficulties of debugging

- “printf” can only be used in “wrapper file”
- If some block has a critical bug, it kills Simulink itself
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
For functional specification, PeaCE uses an extended synchronous dataflow (SDF) model:

- A block is executable as soon as it receives the specified number of samples at all input ports.
- The block produces the specified number of samples at all output ports per each invocation.
- The number of samples consumed (or produced) at each input (or output) port per block invocation is fixed at compile time.
The Schematic of H.264 Decoder in PeaCE

🌟 Clock block is no more needed!
Block Definition in PeaCE

พฤกษ์ทรัพย์

Block definition
- Default program language in PeaCE: C
- Code sections
  - Global declaration, procedure definition, main declaration, and so on
- Each block should follow SDF semantics

Block debugging
- PeaCE generates simulation C codes to simulate system
- This enables all conventional debugging approaches to be used
- If a block has a critical bug, it would kill only the simulation code
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
Simulation Performance of Simulink and PeaCE

Simulation environment
- Application: H.264 decoder
- Input: QCIF (176*144) format motion picture 100 frames
- Host machine
  - CPU: Intel Xeon 1.8Ghz dual
  - RAM: RDRam 1GB
  - OS: Linux Red Hat 7.3

Simulation performance
- Simulink: 581 seconds
- PeaCE: 2.05 seconds
1. Introduction
2. H.264 decoder modeling in Simulink
3. H.264 decoder modeling in PeaCE
4. Simulation performance of Simulink and PeaCE
5. Conclusion
Conclusion

- We present our experience of H.264 decoder algorithm specification and simulation in Simulink and PeaCE and compare them in these three aspects:
  1. Algorithm modeling of multirate system
  2. Block definition and debugging
  3. Simulation performance

- PeaCE can do:
  - Specifying multi-rate DSP systems more easily with formal data-driven model (SDF)
  - System debugging easily and simulating it very fast with automatically generated simulation code
Thank You!!