Dynamic Flow Regulation for IP Integration on Network-on-Chip

Zhonghai Lu and Yi Wang

Dept. of Electronic Systems
KTH Royal Institute of Technology
Stockholm, Sweden

6th Symposium on NoCS, Denmark May 9-11, 2012
Agenda

- The IP integration problem
- Why flow regulation?
- Online flow characterization
- Dynamic regulation
- Experiments and results
- Conclusion and future work
SoC Design

- Design of IPs
  - Separate concerns, e.g. in computation and communication;
  - A divide-conquer approach to manage complexity;
  - by IP vendors

- Integration of IPs
  - via a common interface (AHB, AXI, etc.);
  - by SoC integrators
The IP integration problem

- Separating concerns helps to manage complexity and reuse expert knowledge. However, this creates performance (uncertainty, quality) problem for the IP integration phase.

- Can we control the performance?
Flow regulation

- Do not inject traffic as soon as possible
  - As-soon-as-possible traffic injection creates congestion problem as-soon-as-possible
  - Disciplined traffic helps to alleviate network contention
- A formal foundation: network calculus
  - Abstract flow with arrival curve
  - Abstract server with service curve
- Can be viewed as a **proactive** (vs. **reactive**)
  congestion control scheme

You have the horse.
You have the rein!
An arrival curve $\alpha(t)$ provides an upper bound on the cumulative amount of traffic over time.

A linear arrival curve has the form $\alpha(t) = \sigma + \rho(t)$ where $\sigma$ bounds traffic burstiness, $\rho$ average rate.
Assume: F: Linear arrival curve
\[ \alpha(t) = \sigma + \rho(t) \]

S: Latency-rate server
\[ \beta(t) = R(t - T)^+ \]

- The delay bound is
\[ D = T + \frac{\sigma}{R} \]

- The backlog bound is
\[ B = \sigma + \rho T \]
Why regulation helps?

- Reduce the traffic burstiness
- It in turn reduces contention and buffering requirements in the interconnect.

Example

- Flow without regulation ($\sigma=6.6$, $\rho=0.2$)
- Flow with strongest regulation ($\sigma=1$, $\rho=0.2$)
Online flow characterization

- **Purpose:** Characterize flow’s \((\sigma, \rho)\) values
- **How:** through a sliding window mechanism
  - Calculate previous-window, current-window \((\sigma, \rho)\) values
  - Predict next-window \((\sigma, \rho)\) values
  - The \((\sigma, \rho)\) values are updated window by window
  - The sampling window slides with overlapping, ensuring continuity of predicted values
Online flow characterization

- Sampling window: 750
- Predication window: 250
Sliding window

- Sampling window: 750
- Predication window: 250

\[(\sigma, \rho)\] updates
Sliding window

Sampling window: 750
Predication window: 250
Sliding window

- Sampling window: 750
- Predication window: 250
Sliding window

- Sampling window: 750
- Predication window: 250
Rate $\rho$ characterization

- **Characterize:**
  
  \[ \rho = \frac{f(L_{sw})}{L_{sw}} \]

- **Predict:**
  
  - base value + offset value
  
  \[ \hat{\rho}_{n+1} = \rho_n + (\rho_n - \rho_{n-1}) \]

  - Use history information
  - exploit the continuity brought by the sliding window mechanism to avoid abrupt change
Burstiness $\sigma$ characterization

- **Characterize:**

$$\sigma = f(t_c) - \rho \cdot t_c = f(t_c) - \frac{f(L_{sw})}{L_{sw}} \cdot t_c$$

- **Critical instant, $t_c$**, to calculate a $\sigma$ bound per window

- **Predict:**

$$\hat{\sigma}_{n+1} = \sigma_n + (\sigma_n - \sigma_{n-1})$$
Characterizer in hardware

- Main components: Sampling + Characterize + Predict
  - Sampling \((t, f(t))\)
  - Characterize for current profile \((\sigma, \rho)\)
  - Predict for regulator parameter

- Delay
  - Release the resets with interval of \(L_{pw}\)
  - Overlapping execution => overlapping windows

- MUX
  - Select results and feed them into “Predict”

2 GHz, 12 K NAND gates (45 nm)
Dynamic regulator

- Leaky-bucket regulation mechanism
  - Incoming flow is served only when token is available.
  - Token generate follows a linear curve
- Regulator’s \((\sigma, \rho)\) parameters are fed by the characterizer

Server (1 unit data per token)

\(\sigma\)

Token rate \(\rho\)

Input flow

\((\sigma, \rho)\) regulated flow

1.4GHz, 2.2K NAND gates (45 nm)
Experiments

- Experiment 1: Fidelity of the sliding window based online flow characterization
- Experiment 2: Effect of dynamic flow regulation vs. static regulation vs. no regulation
Experiment 1: Fidelity of characterization

- Build a model for the online characterizer in Matlab
- Use a two-state (on/off) MMP (Markov Modulated Process) as the traffic source
Effectiveness

- Sampling window 8192 cycles, prediction window 2048 cycles.
- Compared to static characterization, dynamic characterization closely reflects the traffic dynamics.
Window overlapping impact

- The Y axis gives the ratio of violation (occasions when real traffic surpasses the projected bound)
- A performance/cost tradeoff: Higher overlapping, lower violation ratio but higher implementation cost.
Experiment 11: Effect of dynamic regulation

- Use RTL models for characterizers, regulators and the network
- The network is a deflection network as it is more challenging to control
- Use both synthetic traffic and Splash2 benchmark traces
Experimental setup

- 56 masters, 8 slaves.
- Measure regulation delay and network delay.
Experimental configuration

Three configurations:

- **No regulation**: Characterizer is disabled, regulator provides a bypass.
- **Static regulation**: Regulators are configured once with offline profiled \((\sigma, \rho)\) values.
- **Dynamic regulation**: Characterizers are enabled. Regulators are dynamically configured.
Synthetic traffic

- 56 masters inject the on-off traffic to 8 slaves with equal probability, creating a hot spot traffic pattern which mimics memory access scenarios.
- Each master generates 8 flows, each targeting a slave. The 8 flows from the same master are treated as 1 aggregate.
Dynamic regulation outperforms static regulation for 34 (61%) of the 56 aggregates, with the maximum and average reduction of 452 cycles (16%) and 146.8 cycles (5.8%).

Dynamic regulation outperforms no-regulation for 46 (82%) of the 56 aggregates. The maximum and average improvement is 435 cycles (17.4%) and 167.5 cycles (6.3%).
Dynamic regulation outperforms static regulation for all 56 aggregates, with the maximum and average reduction of 186 cycles (13.8%) and 108.6 cycles (14.5%), resp.

Dynamic regulation outperforms no-regulation for 45 (80%) of the 56 aggregates. The maximum and average improvement is 332.8 cycles (54.6%) and 147.8 cycles (17.7%), resp.
Splash2 benchmark traces

- Full-system simulator SIMICS together with GEMS (for the memory system).
- According to the figure, we configured a CMP system with 56 cores (masters) and 8 slaves.
- Each core has L1 I/D Caches: 64KB, 4 way set-associative; L2 Cache: 256KB, 4 way set associative, 64 Byte lines.
- Total off-chip memory size is 4 GB with each memory being 500 MB (4G/8).
- Directory-based MOESI protocol.
- The configured CMP system runs Solaris 9 OS.
- After being compiled, the benchmark programs ran on the OS and traces were recorded.
Compared to static regulation, the improvement in overall average packet delay ranges from 12 to 90 cycles, from 10% to 26% in percentage.

Compared to no-regulation, it is from 53 to 190 cycles, from 22% to 41% in percentage.
Conclusion

- Online traffic profiling through a sliding window presents good fidelity and enables efficient hardware implementation.

- Integrating the online characterization into flow regulation enables dynamic proper adjustment of regulation strength.

- Compared to static and no regulation, dynamic regulation is more powerful in improving maximum and average packet delay.
When delay is reduced?

- Delay reduction of dynamic vs. static regulation for FFT

Future work: include network status into the control loop.
Acknowledgements

Thanks for your attention!