The Gannet Service-based SoC: A Service-level Reconfigurable Architecture

Wim Vanderbauwhede

Department of Computing Science

University of Glasgow, UK

Overview

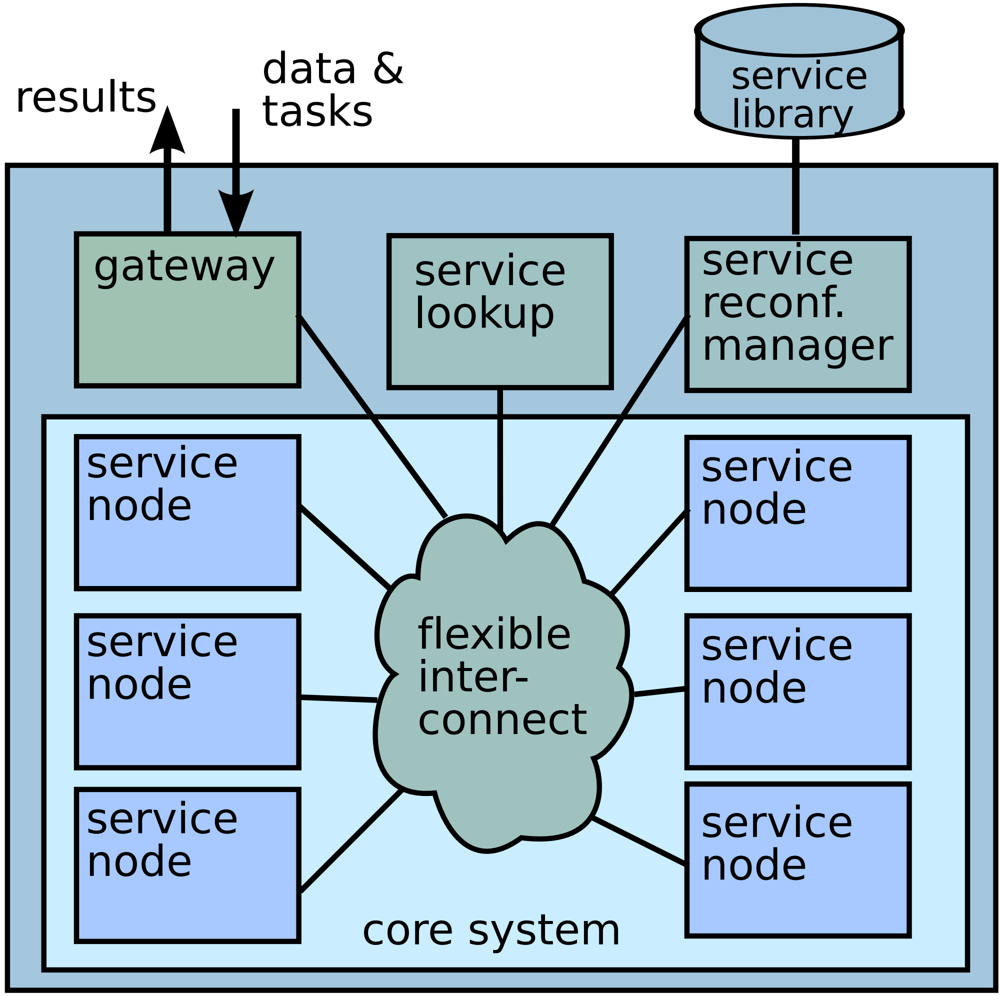
- Overview of the Gannet architecture
- Operation principle
- Gannet task descriptions
- Service manager design
- Performance improvement through control services
- Conclusion

Architecture overview

- a **service-based** architecture for **very large Systems-on-Chip**
  - a collection of processing cores (HW/SW)
  - each core offers a a specific **service**
  - all services are **fully connected** over an on-chip network (NoC)
  - all information is transfered as **packets** over the NoC
- task-level reconfigurability
- high abstraction-level design

Architecture overview

- a **service-based** architecture for **very large SoCs**

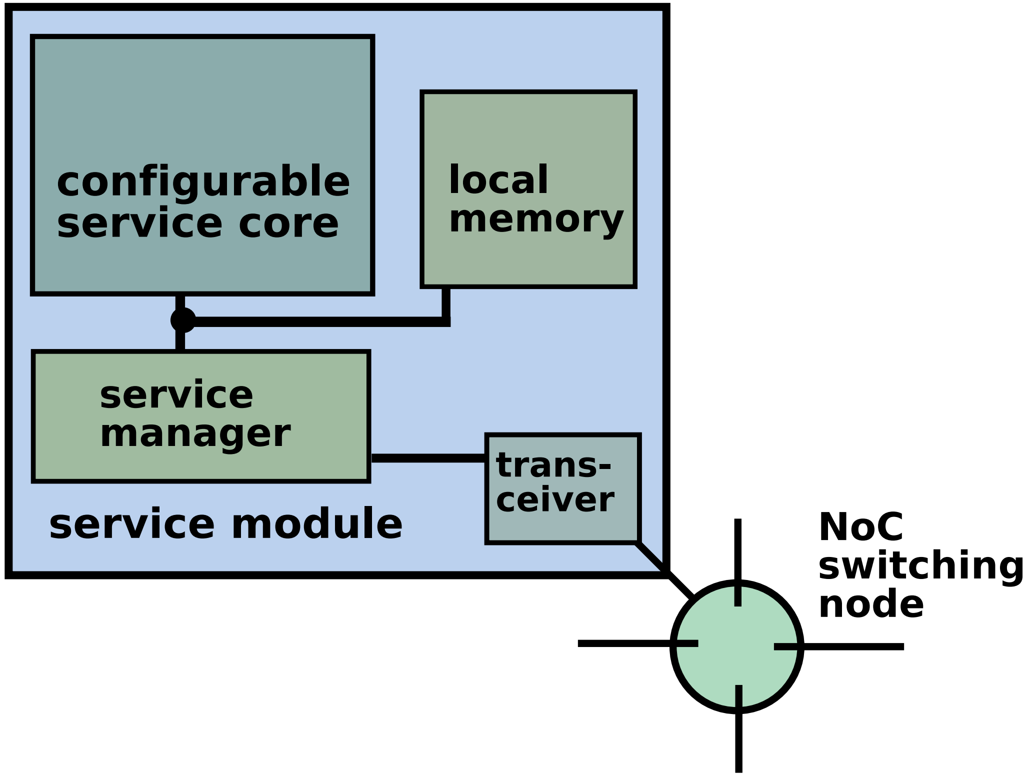


Operation principle

- the SoC’s services collaborate in a demand-driven dataflow fashion:
  - **data** enter the system
  - to be processed by **services**
  - the **results** of which are, like the data, processed by services
  - this process evolves according to a predefined but configurable **task**
  - the **description** of such a task is a Gannet **program**
- the SoC does **not** require a central controller

Managing the service dataflows

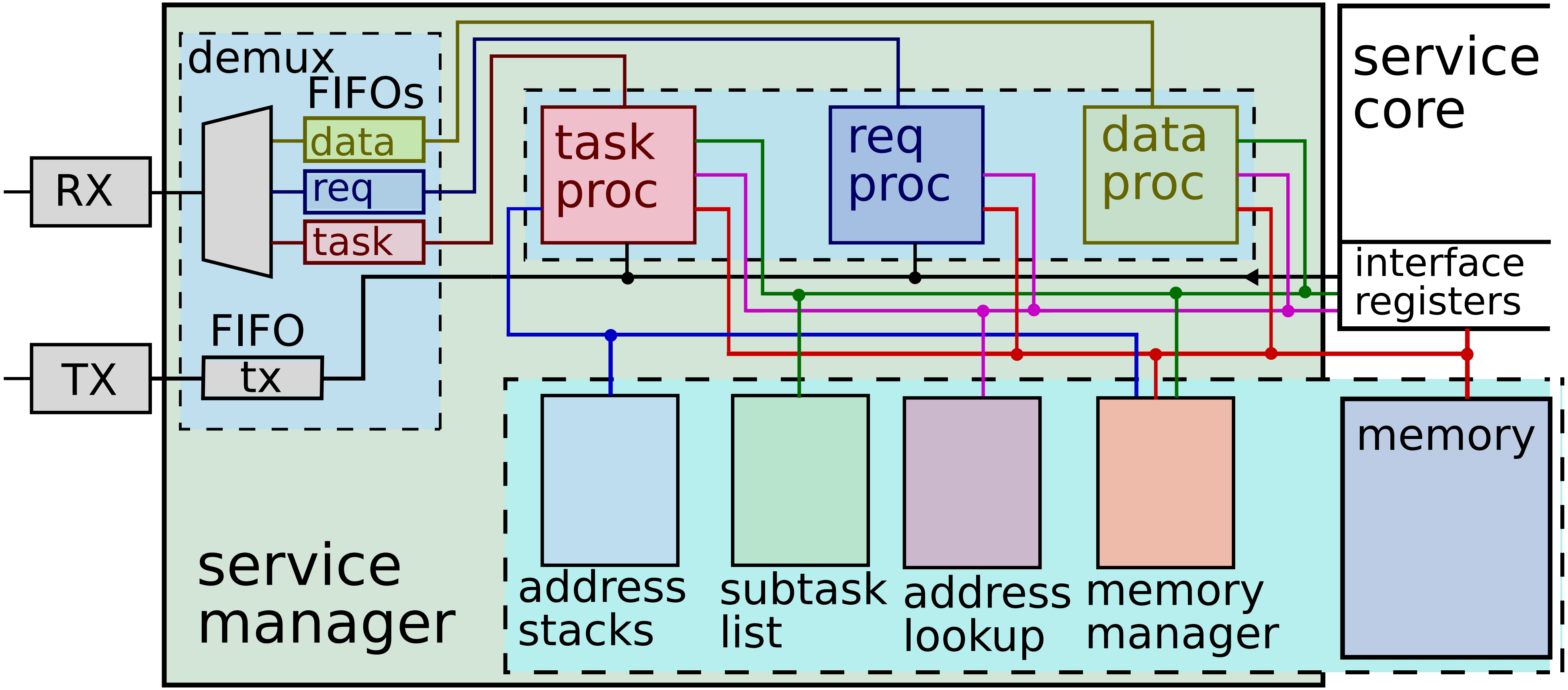
- to manage the flow of **data** and **task descriptions** between the **heterogenous service cores**, **every** core interfaces with the system through a **service manager**



Managing the service dataflows

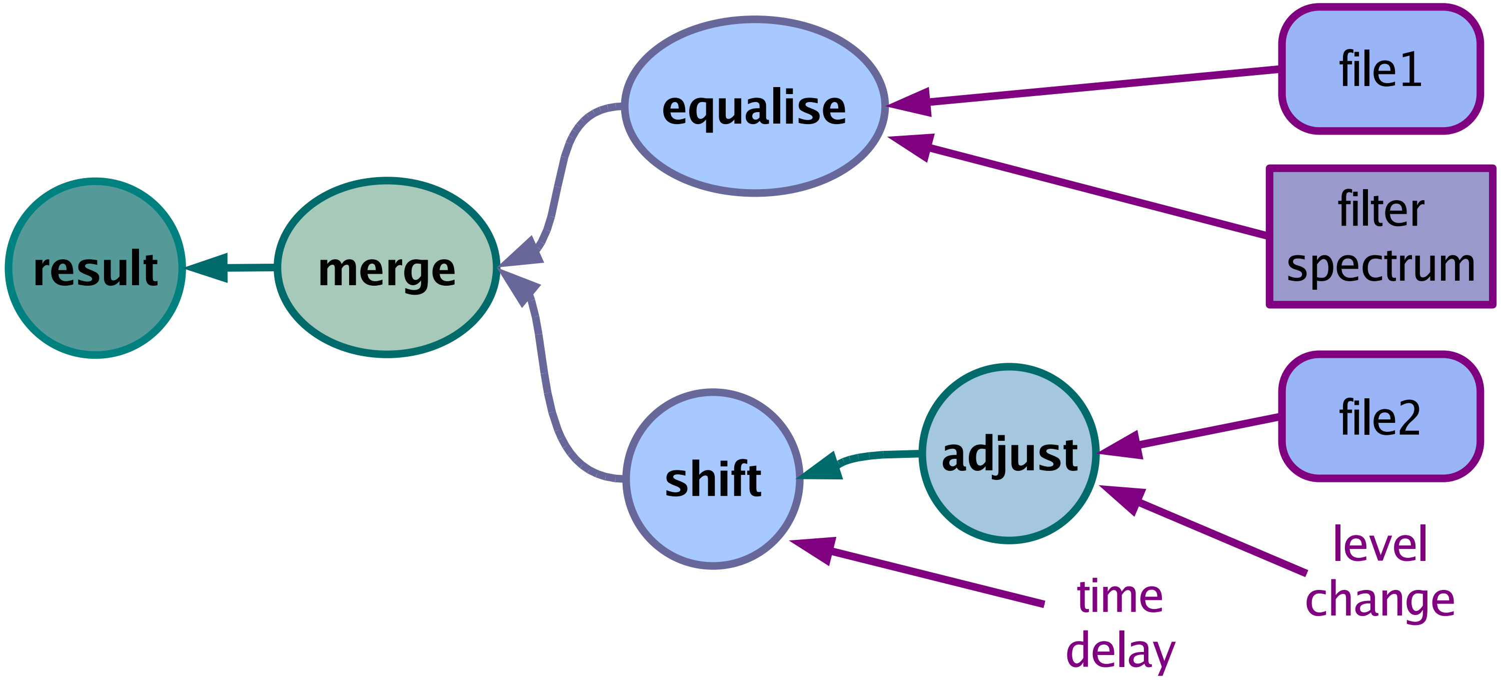
- the task description is a list of **symbols** (64-bit words) representing either **data** or **tasks**
- essentially, the service manager uses two rules to evaluate the task description:
  - data ⇒ request
  - task ⇒ delegate
- it keepts track of all pending subtasks and the status of the data required by them
- the service cores are task-agnostic

Service manager design



Task descriptions

example task: a system to process audio files.



Task description syntax

example (C-like syntax)

```
float time_delay,level_change;
Audiofile* file1,file2;
Spectrum* filter_spectrum;

merge(
    shift(time_delay,
        adjust(level_change,file2)),
    equalise(filter_spectrum,file1));
```

Performance improvement through control services

the Gannet service-based architecture:

- allows to describe and execute arbitrary complex tasks:
  - transparent interaction between cores
  - concurrency by design
  - no race conditions
- but has room for improvement:
  - memory requirements
  - limited parallelism – no fan-in
  - no conditional branching
  - no loop constructs – program size

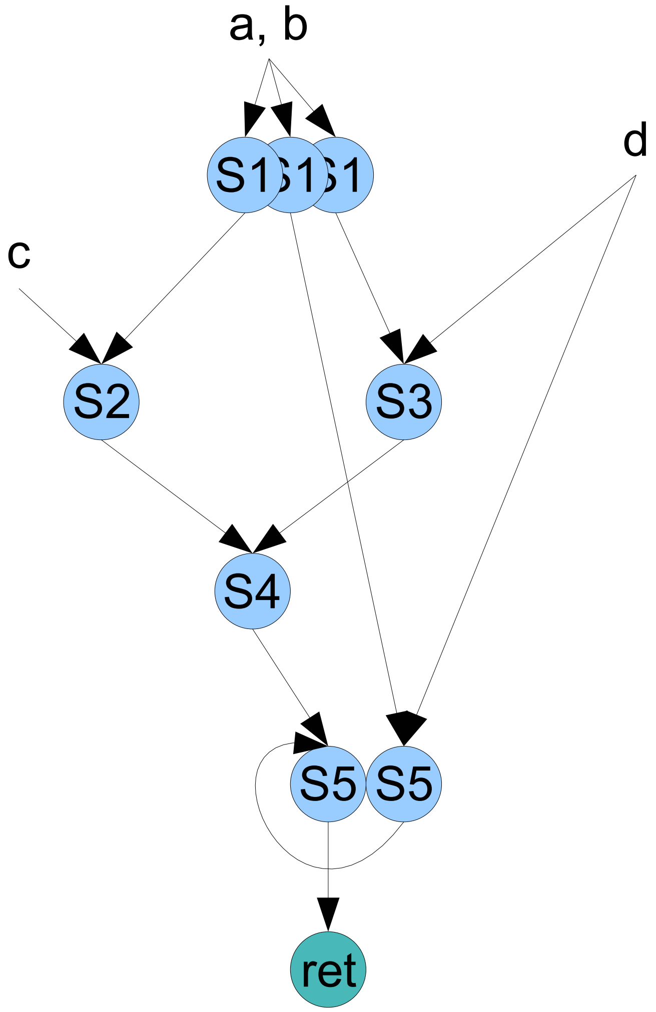
Performance improvement through control services

control services:

- services that add specific control functionality to the system
  - variables: store results
  - conditional branching
  - memory control
  - subroutines
  - parallelism

Variables

- Task without variables



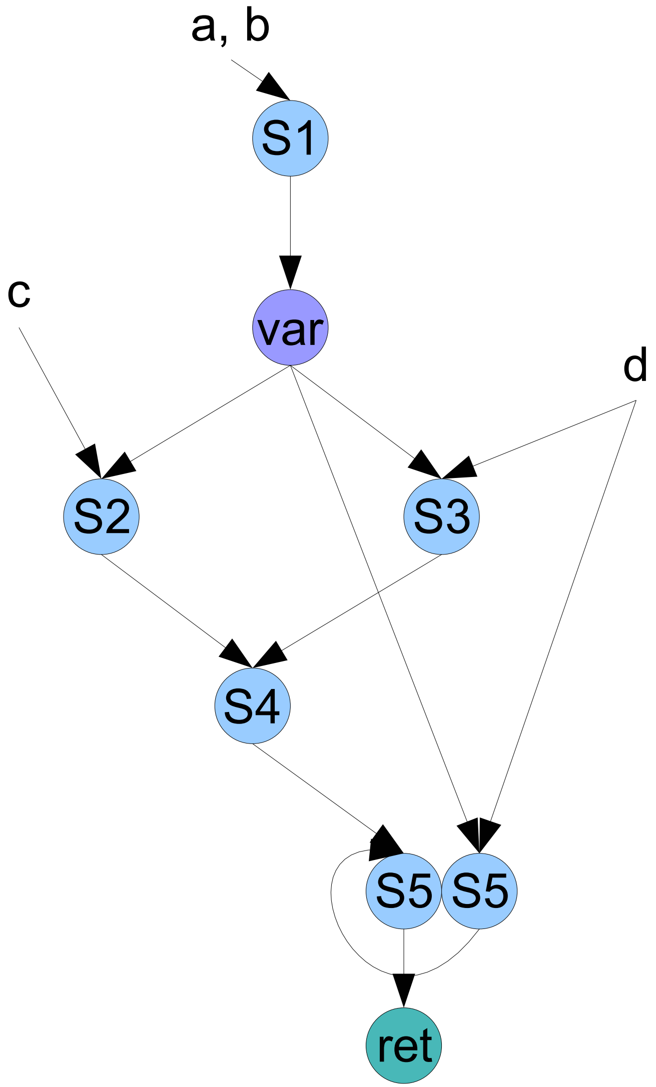
```
data* a,b,c,d;

return S5(
    S4(
        S2(S1(a,b),c)
        S3(S1(a,b),d)
        ),
    S5(S1(a,b),d)
    );
```

// S1(a,b) gets calculated 3 times

Variables

- Task with variables

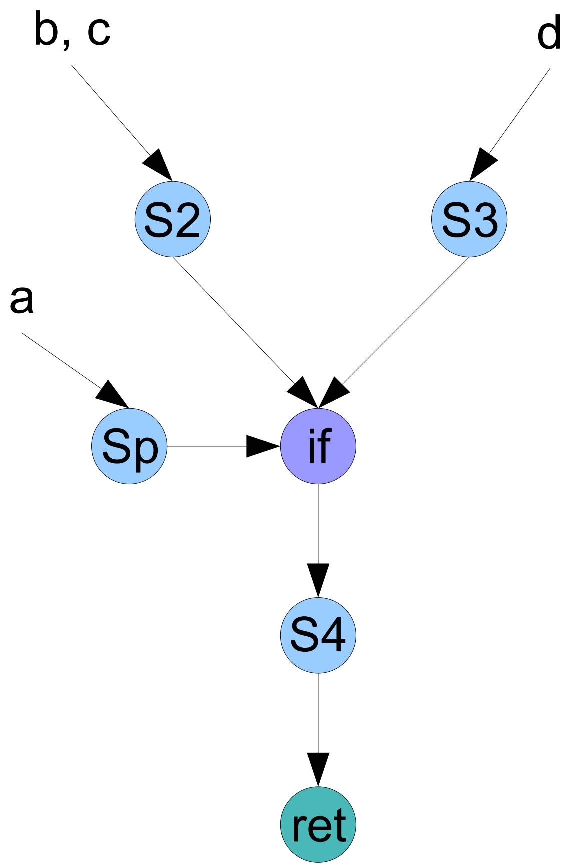


```
data* a,b,c,d;

v=S1(a,b);
return S5(
    S4(
        S2(v,c)
        S3(v,d)
        ),
    S5(v,d)
    );
```

// => S1(a,b) gets calculated once

Conditional branching

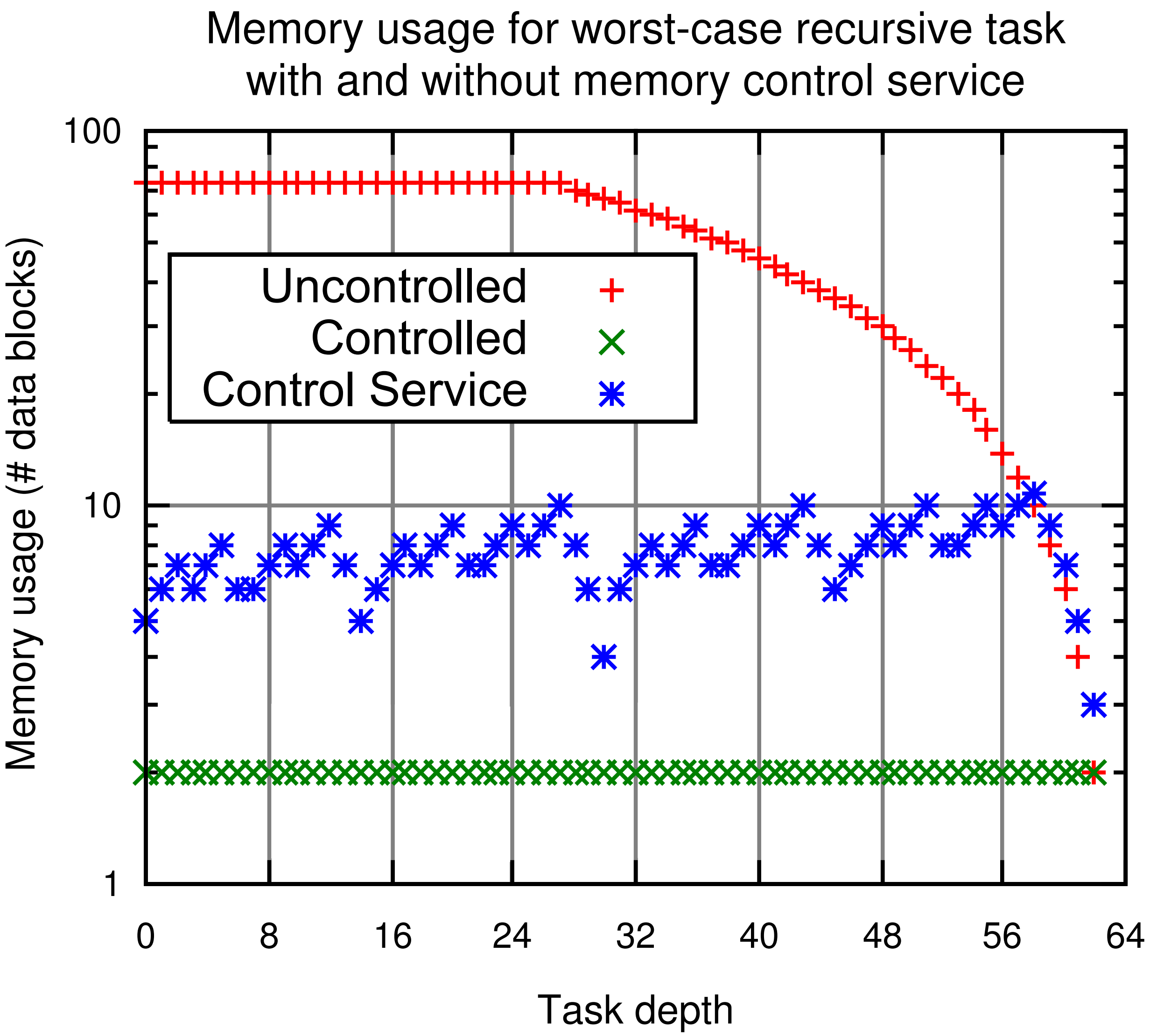


// Branching service: if

```
data* a,b,c,d;

return S4(
    if(Sp(a),
        S2(b,c),
        S3(d)
    )
);
```

Example: memory usage



Conclusion

- Gannet project: facilitate high abstraction-level design of complex SoCs
- Novel **service-based** SoC architecture: IP cores are service providers
- Distributed processing system – no central control, full concurrency
- Service manager for transparent interaction between cores
- High-level **task description** language
- Introducing **control services** to improve system performance