TDTD30: Design space exploration

Outline

§ Introduction
§ Design Flow
§ Design Space Exploration
§ Communication in MPSoCs
§ Mapping & Scheduling in MPSoCs
Administrative Info

- Labs in the SU rooms
- ssh to a remote Linux computer from the pool (sy00-1.ida.liu.se – sy007.ida.liu.se)

**Ex:**

```bash
ssh su00-3.ida.liu.se
```

```bash
source /home/TDTS30/sw/mparm/go_mparm.sh
```

---

**GSM Phone:**

- Search
- Radio Link Control
- Talking

**MP3 player**

**Digital Camera:**

- Take Photo
- Restore Photo

... High performance

... Time to market

... Low power
MPSoC Architecture

System Design Flow
MPARM: Hardware

- Simulation platform
- Up to 8 ARM7 processors
- Variable frequency (dynamic and static)
- Split/Unified instruction and data caches
- Private memory for each processor
- Scratchpad
- Shared Memory
- Bus: AMBA, ST

This platform can be fine tuned for a given application.

This platform can be fine tuned for a given application. Read more in:
/home/TDTS30/sw/mparm/MPARM/doc/simulator_statistics.txt
MPARM: Software

- C crosscompiler chain for building the software applications
- No operating system
  - Small set of functions (pr, shared_alloc, WAIT/SIGNAL)
- RTEMS operating system

MPARM: Why?

- Cycle accurate simulation of the system
- Various interesting statistics: number of clock cycles executed, bus utilization, cache efficiency, energy/power consumption of the components (CPU cores, bus, memories)
**MPARM: How?**

- `mpsimm.x –c2` runs a simulation on 2 processors, collecting the default statistics.
- `mpsimm.x –c2 –w` runs a simulation with power/energy statistics.
- `mpsimm.x –c1 –is 9 –ds 10 :one processor with instruction cache of size 512 and data cache of 1024 bytes`.
- `mpsimm.x –c2 –F0,2 –F1,1 –F3,3 :two processors running at 100 MHz, 200 MHz and the bus running at 66 MHz`.
- `mpsimm.x –h` for the rest.

Simulation results are in the file `stats.txt`.

---

**Design Space Exploration**

- **Generic term for system optimization**
- **Platform optimization:**
  - Select the number of processors
  - Select the speed of each processor
  - Select the type, associativity and size of the cache
  - Select the bus type
- **Application optimization**
  - Select the interprocessor communication style (shared memory or distributed message passing)
  - Select the best mapping and schedule
Design Space Exploration for the GSM codec

Assignment 1

- Given the GSM code
- Running on 1 ARM7 processor
- The variables are:
  - cache parameters
  - processor frequency
- Using the MPARM simulator, find a hardware configuration that minimizes the energy of the system

Energy/Speed Tradeoff

- 0.75V, 60mW, 150MHz
- 1.3V, 450mW, 600MHz
- 1.6V, 900mW, 800MHz

- IDLE: 40mW, 10us, 90us
- SLEEP: 160mW, 140ms, 1.5ms
- RUN: 0.75V, 60mW, 10us, 160us
Frequency Selection: Total Energy

![Graph showing energy consumption vs frequency divider for Total Energy.]

Frequency Selection: ARM Core Energy

![Graph showing energy consumption vs frequency divider for ARM Core Energy.]

Instruction Cache Size: Total Energy

Energy [mJ]

log2(CacheSize)

$2^9=512$ bytes

$2^{14}=16$ kbytes

Instruction Cache Size: Execution Time

t [cycles]

log2(CacheSize)
Interprocessor Data Communication

CPU₁
...
\( a = 1 \)
...

CPU₂
...
print \( a; \)
...

BUS

Shared Memory

CPU₁
...
\( a = 1 \)
...

CPU₂
\( a = 2 \)
print \( a; \)

a=?

Shared Mem

a

BUS

Synchronization
Synchronization

CPU₁
\[ a = 1 \]
\[ \text{signal(sem}_a \text{)} \]

Semaphore
\[ \text{sem}_a \]

CPU₂
\[ a = 2 \]
\[ \text{wait(sem}_a \text{)} \]
\[ \text{print a;} \]

Synchronization In[f/t]ernals

CPU₁
\[ a = 1 \]
\[ \text{signal(sem}_a \text{)} \]

Semaphore
\[ \text{sem}_a \]

CPU₂
\[ a = 2 \]
\[ \text{wait(sem}_a \text{)} \]
\[ \text{print a;} \]

while \[ (\text{sem}_a === 0) \]

Shared Mem
\[ a \]

sem_a=1

polling

BUS
Distributed Message Passing

CPU₁

a = 1

signal(sem_a)

CPU₂

wait(sem_a)

a = 2

print a;

sem_a

Shared Mem

a

BUS

Distributed Message Passing (2)

CPU₁ (prod)

a = 1

signal(sem_a)

CPU₂ (cons)

wait(sem_a)

print a;

sem_a

a = 1

BUS
Assignment 2

- You are given 2 implementations of the GSM codec
  - Shared memory
  - Distributed message passing
  - Simulate and compare these 2 approaches

System Design Flow

1. Hardware platform
2. Software Application(s)
3. Extract Task Graph
4. Extract Task Parameters
5. Optimize (Mapping & Sched)
6. Implement
7. Formal Simulation
Task Graph Extraction Example

for (i=0; i<100; i++) a[i]=1;//TASK 1
for (i=0; i<100; i++) b[i]=1;//TASK 2
for (i=0; i<100; i++) c[i]=a[i]+b[i];//TASK 3

Task 1 and 2 can be executed in parallel
Task 3 has data dependency on 1 and 2

Execution Time Extraction

§ Using the simulator
§ This an "average" execution time
§ In real-time systems, another execution
time is used: worst-case (WCET)
§ Can be extracted using the
dump_light_metric(int x) in MPARM
Execution Time Extraction Example

```c
start_metric();
for (i=0; i<100; i++) a[i]=1; // TASK 1
dump_light_metric(1);
for (i=0; i<100; i++) a[i]=1; // TASK 2
dump_light_metric(2);
stop_metric();
stop_simulation();
```

---

### Execution Time Extraction Example (2)

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Interconnect statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall exec time</td>
<td>287 system cycles (1435 ns)</td>
</tr>
<tr>
<td>Task NC</td>
<td>287</td>
</tr>
<tr>
<td>1-CPU average exec time</td>
<td>0 system cycles (0 ns)</td>
</tr>
<tr>
<td>Concurrent exec time</td>
<td>287 system cycles (1435 ns)</td>
</tr>
<tr>
<td>Bus busy</td>
<td>144 system cycles (50.17% of 287)</td>
</tr>
<tr>
<td>Bus transferring data</td>
<td>64 system cycles (22.30% of 287, 44.44% of 144)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 2</th>
<th>Interconnect statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall exec time</td>
<td>5554 system cycles (27770 ns)</td>
</tr>
<tr>
<td>Task NC</td>
<td>5267</td>
</tr>
<tr>
<td>1-CPU average exec time</td>
<td>0 system cycles (0 ns)</td>
</tr>
<tr>
<td>Concurrent exec time</td>
<td>5554 system cycles (27770 ns)</td>
</tr>
<tr>
<td>Bus busy</td>
<td>813 system cycles (14.64% of 5554)</td>
</tr>
<tr>
<td>Bus transferring data</td>
<td>323 system cycles (5.82% of 5554, 39.73% of 813)</td>
</tr>
</tbody>
</table>
Application Mapping and Scheduling

Mapping in MPARM

- Using the RTEMS OS -> not this time
- Using the lightweight API of MPARM
  - get_proc_id()
Mapping in MPARM Example

if (get_proc_id() == 1) {
    // Task 1 executed on processor 1
    for (i = 1; i < 100; i++) a[i] = 1;
}
if (get_proc_id() == 2) {
    // Task 1 executed on processor 2
    for (i = 1; i < 100; i++) b[i] = 1;
}

Scheduling in MPARM

$\text{Using the RTEMS OS -> not this time}$

$\text{The schedule is given by the code sequence executed on one processor}$

// task 1
for (i = 1; i < 100; i++) a[i] = 1;
// task 2
for (i = 1; i < 100; i++) b[i] = 3;

Schedule 1

Schedule 2

// task 2
for (i = 1; i < 100; i++) b[i] = 3;
// task 1
for (i = 1; i < 100; i++) a[i] = 1;
Assignment 3: Mapping and Scheduling

- GSM codec
- Task graph is given
- Extract the task execution times
- Given a mapping, improve the scheduling
- Find a better mapping