There are n persons. All have one secret to tell, which is not known to the others. They are very desperate to tell their secrets and hear what the other persons know. They communicate via a two-way channel (e.g., a telephone); thus, when two persons are on the phone, they exchange the secrets they currently know. What is the minimum telephone calls needed so that each person knows all secrets?

- A person should be able to call and answer (channels)
- How do we represent secrets? (integer variable)
- How do we communicate secrets? (global variable)
Lab 3: Design space exploration in MPARM
Lab 3 - The MPARM simulation framework

- HW platform
- SW platform
- Design space exploration for energy minimization
- Communication in MPSoC
- Mapping and scheduling
MPSoC Architecture

ARM
CACHE

ARM
CACHE

ARM
CACHE

Interrupt Device

Private Memory

Private Memory

Private Memory

Semaphore Device

Shared Memory

Bus (AMBA, ST)
System Design Flow

Informal specification, constraints

Modeling

Architecture selection

System architecture

Estimation

Mapping

Scheduling

Mapped and scheduled model

Hardware and Software Implementation

Testing

Prototype

Fabrication

not ok

Functional simulation

not ok

ok
System Design Flow

Hardware platform

Software Application(s)

Extract Task Graph

Extract Task Parameters

Optimize (Mapping & Sched)

Formal Simulation

Implement
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

Read more in:
/home/TDTS07/sw/mparm/MPARM/doc/simulator_statistics.txt
- C crosscompiler chain for building the software applications
- No operating system
  - Small set of functions (pr, shared_alloc, WAIT/SIGNAL)
  - (look in the application code)
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)
- `mpsim.x -c2` runs a simulation on 2 processors, collecting the default statistics
- `mpsim.x -c2 -w` runs a simulation with power/energy statistics
- `mpsim.x -c1 --is=9 --ds=10` : one processor with instruction cache of size 512 and data cache of 1024 bytes
- `mpsim.x -c2 -F0,2 -F1,1 -F3,3` : two processors running at 100 MHz, 200 MHz and the bus running at 66 MHz
  - 200 MHz is the “default” frequency
- `mpsim.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
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

CPU model

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

- **IDLE**
  - 40mW

- **SLEEP**
  - 160μW

Transition times:
- 160μs
- 10μs
- 10μs
- 140ms
- 90μs
- 1.5ms
Frequency Selection: ARM Core Energy

![Graph showing energy consumption against frequency divider]

- Energy [mJ] on the y-axis
- Freq. divider on the x-axis

The graph illustrates the relationship between energy consumption and frequency divider, showing a significant decrease in energy as the frequency divider increases.
Frequency Selection: Total Energy

Energy [mJ]

Freq. divider
Instruction Cache Size: Total Energy

![Graph showing the relationship between energy and cache size.](image)

- **Energy [mJ]**
- **Log2(CacheSize)**
- **2^9 = 512 bytes**
- **2^14 = 16 kbytes**
Instruction Cache Size: Execution Time

![Graph showing the relationship between log2(CacheSize) and execution time (t [cycles]). The graph indicates a decrease in execution time as the log2(CacheSize) increases.]
Interprocessor Data Communication

CPU₁
...

a = 1
...

CPU₂
...

print a;
...

BUS

How?
CPU_1
...
\textbf{a}=1
...

CPU_2
\textbf{a}=2
print \textbf{a};

\textbf{a}=?

\textbf{Synchronization}

Shared Mem

\textbf{a}

BUS
With semaphores

Synchronization

CPU_1

\[ a = 1 \]

signal(sem_a)

Semaphore

\[ \text{sem}_a \]

CPU_2

\[ a = 2 \]

wait(sem_a)

print \( a \);

Shared Mem

a

BUS
Synchronization In[f/t]ernals

CPU₁
a = 1
signal(sem_a)

CPU₂
while (sem_a == 0)
wait(sem_a)
a = 2
print a;

sem_a = 1

Semaphore
sem_a

Shared Mem
a

BUS

polling
Disadvantages of polling:

- Results in higher power consumption (energy from the battery)
- Larger execution time of the application
- Blocking important communication on the bus
Distributed Message Passing

- Instead:
  - Direct CPU-CPU communication with distributed semaphores
  - Each CPU has its own scratchpad
    - Smaller and faster than a RAM
    - Smaller energy consumption than a cache
    - Put frequently used variables on the scratchpad
    - Cache controlled by hardware (cache lines, hits/misses, ...)
    - Scratchpad controlled by software (e.g., compiler)
  - Semaphores allocated on scratchpads
  - → No polling
Distributed Message Passing

CPU₁

\[ a = 1 \]

signal(sem_a)

CPU₂

\[ a = 2 \]

wait(sem_a)

print a;

sem_a

Shared Mem

\[ a \]

BUS
Distributed Message Passing (2)

CPU_1 (prod)
- a = 1
- signal(sem_a)

CPU_2 (cons)
- wait(sem_a)
- print a;

BUS

sem_a
- a = 1
- a = 1
- a = 1
You are given 2 implementations of the GSM codec

- Shared memory
- Distributed message passing

Simulate and compare these 2 approaches

- Energy
- Runtime
System Design Flow

Hardware platform -> Extract Task Graph

Software Application(s) -> Extract Task Graph

Extract Task Parameters

Optimize (Mapping & Sched) -> Implement

Formal Simulation

27
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 is an "average" execution time
- Can be extracted using the `dump_light_metric()` in MPARM
Execution Time Extraction Example

start_metric();
for (i=0;i<100;i++) a[i]=1;//TASK 1
dump_light_metric();
for (i=0;i<100;i++) b[i]=1;//TASK 2
dump_light_metric();
stop_metric();
stop_simulation();
Task 1
Interconnect statistics
-----------------------
Overall exec time = 287 system cycles (1435 ns)
Task NC = 287
1-CPU average exec time = 0 system cycles (0 ns)
Concurrent exec time = 287 system cycles (1435 ns)
Bus busy = 144 system cycles (50.17% of 287)
Bus transferring data = 64 system cycles (22.30% of 287, 44.44% of 144)
-----------------------
Task 2
Interconnect statistics
-----------------------
Overall exec time = 5554 system cycles (27770 ns)
Task NC = 5267
1-CPU average exec time = 0 system cycles (0 ns)
Concurrent exec time = 5554 system cycles (27770 ns)
Bus busy = 813 system cycles (14.64% of 5554)
Bus transferring data = 323 system cycles (5.82% of 5554, 39.73% of 813)
Application Mapping and Scheduling
Mapping in MPARM Example

- Using the lightweight API of MPARM
  - `get_proc_id()`

```c
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;
}
```
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

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

Schedule 2
Assignment 3: Mapping and Scheduling

- Theoretical exercise (you will not use MPARM)
- Task graph and task execution times are given
- Construct 2 schedules with (same) minimal length:
  - keeping same task mapping
  - changing the mapping
Thank you!
Questions?