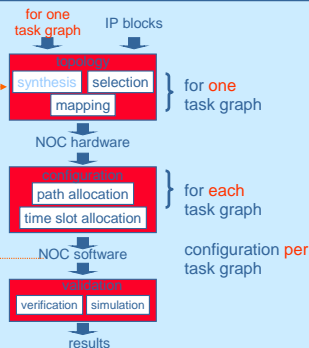
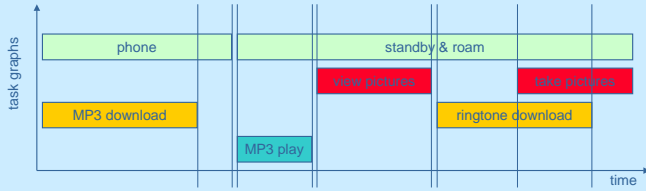


Multiple Use-Cases for a NOC Design Flow

Kees Goossens, IC Design, Philips Research

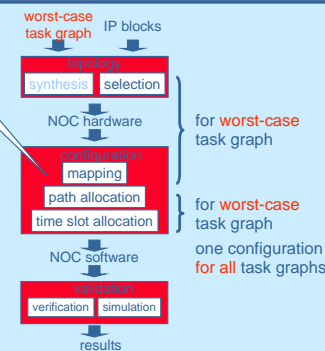
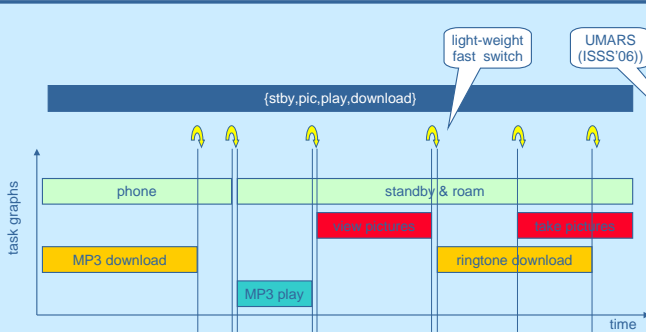
poster is based on CODES+ISSS'05 (Hansson et al.), ASPDAC'06 (Murali et al.), and DATE'06 (Murali et al.)



simple approach

1. generate NOC hardware & software for **one task graph**
2. configure all remaining task graphs
3. **try bigger NOC** if some task graphs fail

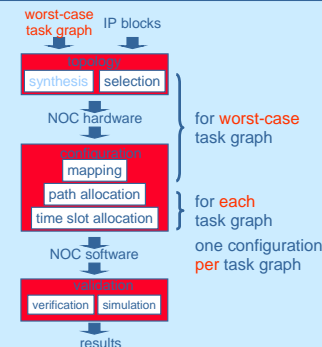
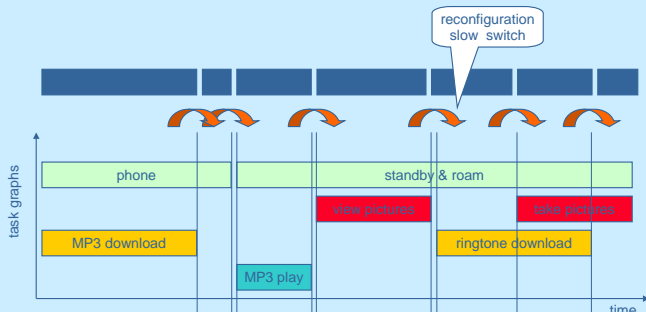
- convergence?



better approach A

1. generate the **worst-case task graph** (all connections simultaneously)
2. generate NOC hardware & software for worst-case task graph
3. use **same configuration** for all task graphs

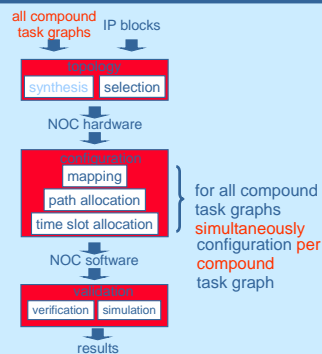
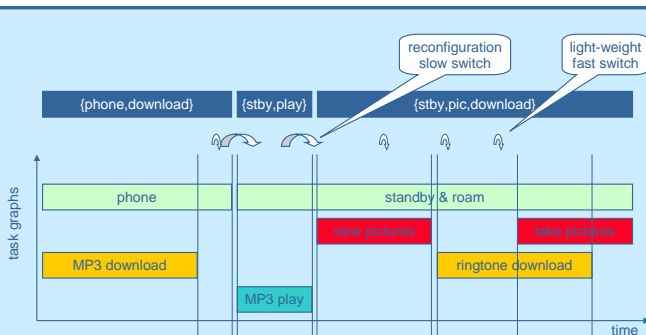
- converges
- **fastest switching**
- highest area, highest frequency



better approach B

1. generate the **worst-case task graph** (all connections simultaneously)
2. generate NOC hardware & software for worst-case task graph
3. **recompute new configuration** for each (compound) task graph

- converges (using A as fall-back)
- slower switching
- highest area, **lower frequency**



best approach

1. (generate all compound task graphs)
2. generate NOC hardware & software for **all (compound) task graphs simultaneously** (by picking worst connection across all task graphs)
3. obtain **configuration** for each compound task graph

- converges
- **switching as needed**
- **lowest area, lowest frequency**

use cases / task graphs / modes

- many modes
- many sub-applications
- interleaving
- transitions

future work

- specification of transition behaviour
- spec. of offered & required resources
- from design time to run time
- resources beyond communication