A Hierarchical Multiprocessor Scheduling System for DSP Applications

José Luis Pino[†]

Shuvra S. Bhattacharyya[‡]

Edward A. Lee[†]

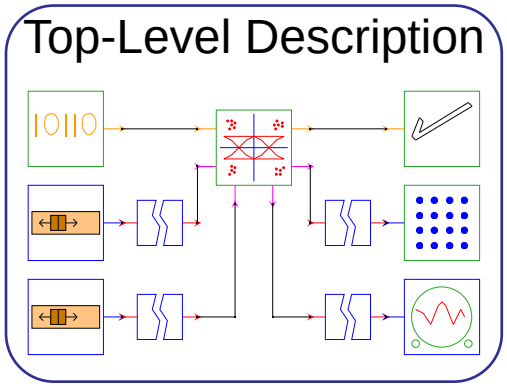
[†]*University of California, Berkeley*

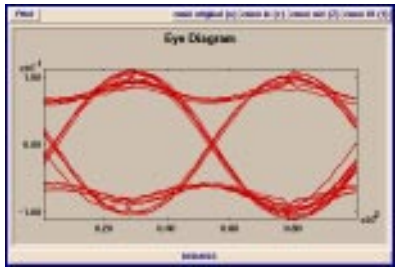
[‡]*Semiconductor Research Laboratory, Hitachi America, Ltd*

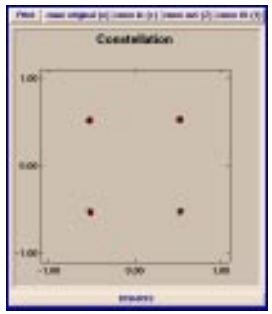
The Ptolemy Project

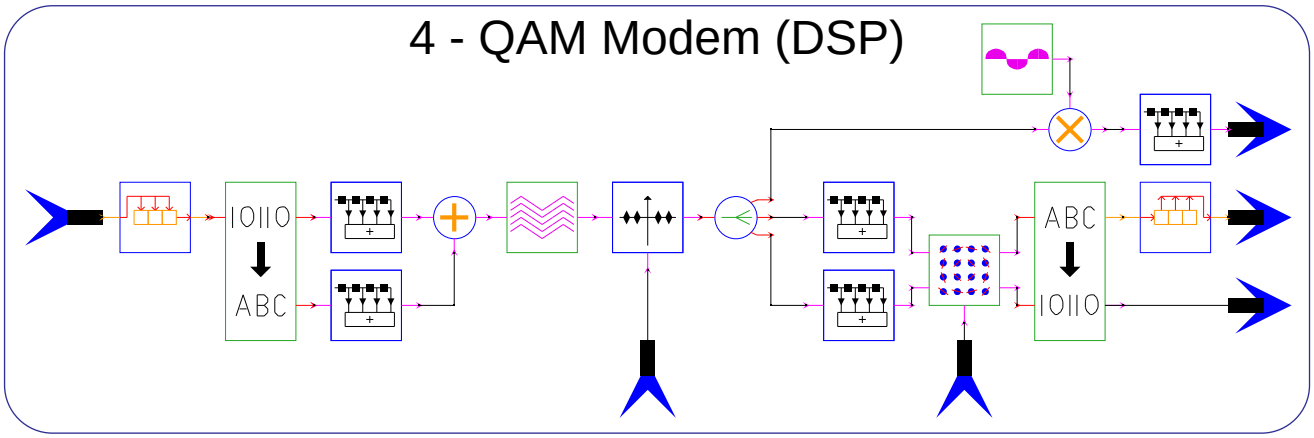
Shuvra Bhattacharyya
Joseph T. Buck
Wan-Teh Chang
Michael J. Chen
Brian L. Evans
Soonhoi Ha
Paul Haskell
Chih-Tsung Huang
Wei-Jen Huang
Christopher Hylands
Asawaree Kalavade
Alan Kamas
Allen Lao
Edward A. Lee
Seungjun Lee
David G. Messerschmitt
Praveen Murthy
Thomas M. Parks
José Luis Pino
S. Sriram
Michael C. Williamson
Kennard White

Design of Reactive & Real-Time Systems





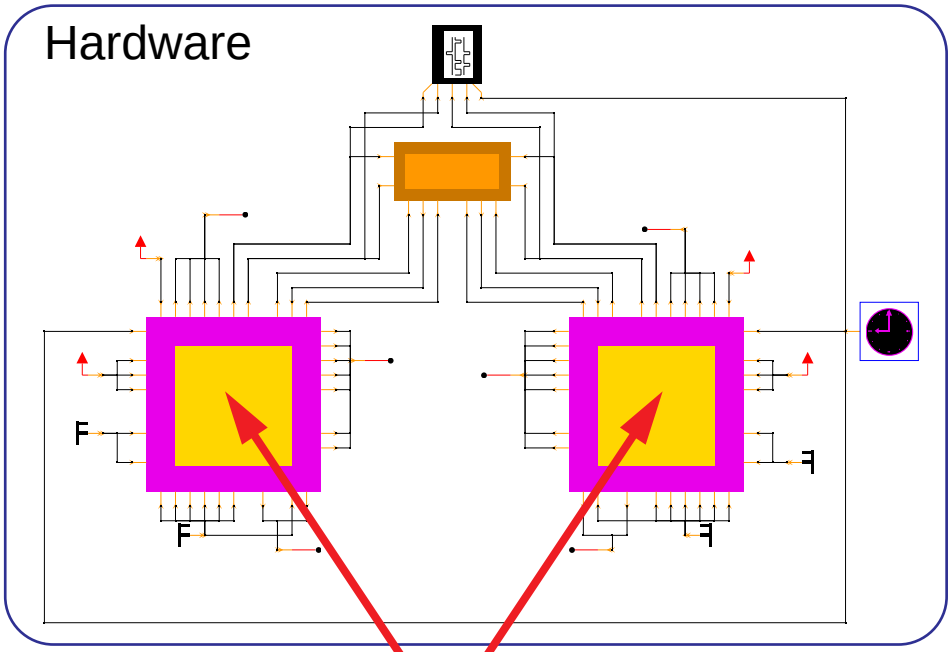


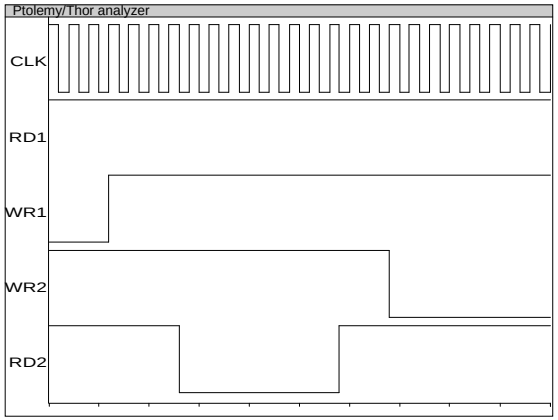


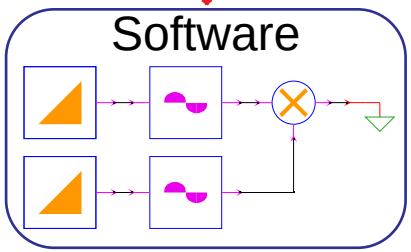
Application designed by José Luis Pino

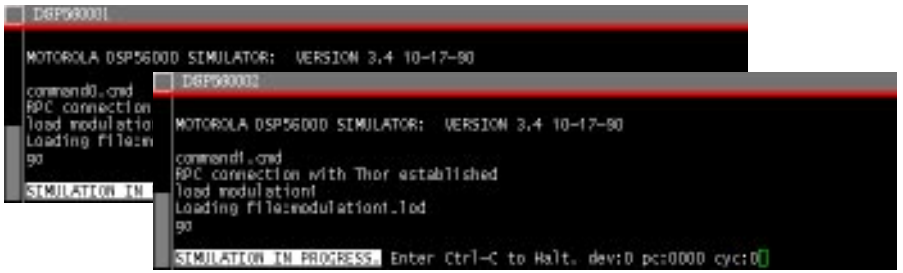
This highly multidisciplinary project addresses system-level design and implementation of reactive and real-time systems.

Implementation Technology





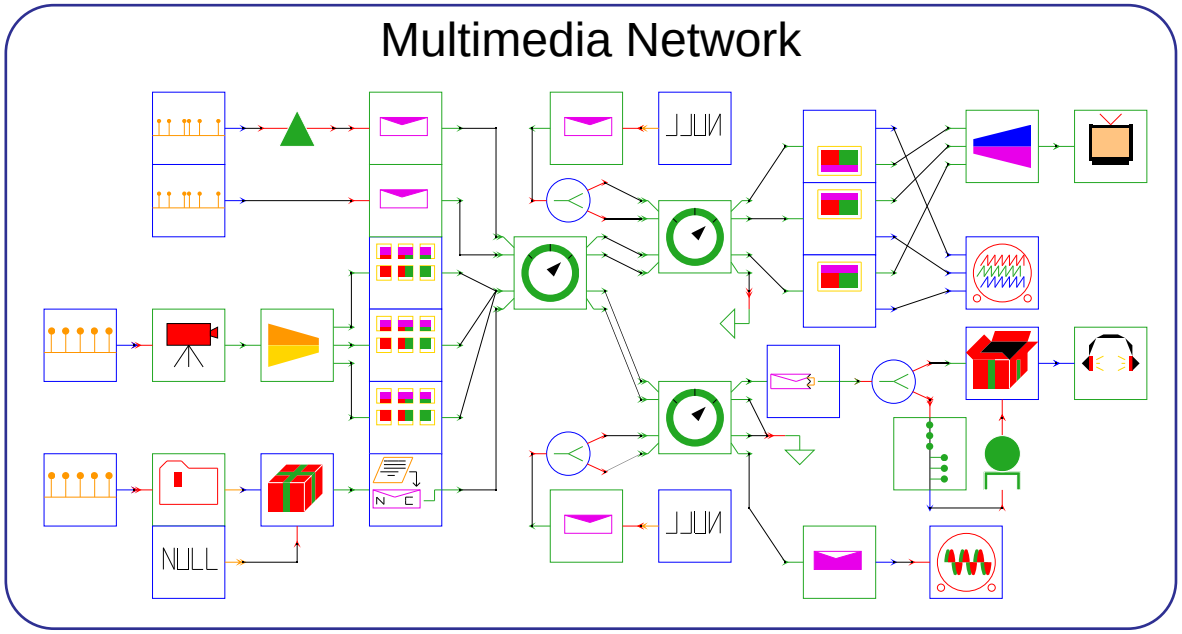


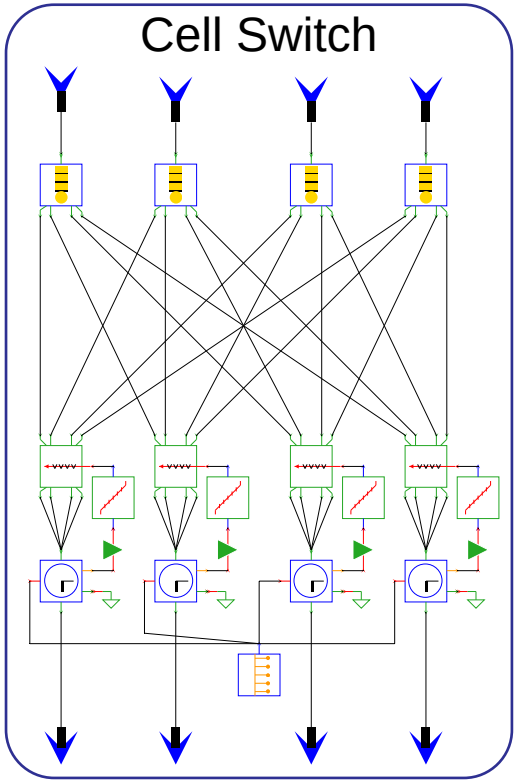


The design philosophy in Ptolemy is heterogeneous, allowing for effective use of specialized design tools within a general system-level design environment.

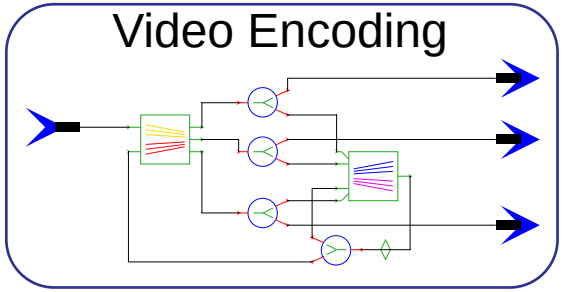
Application designed by Asawaree Kalavade

Heterogeneous Modeling and Design



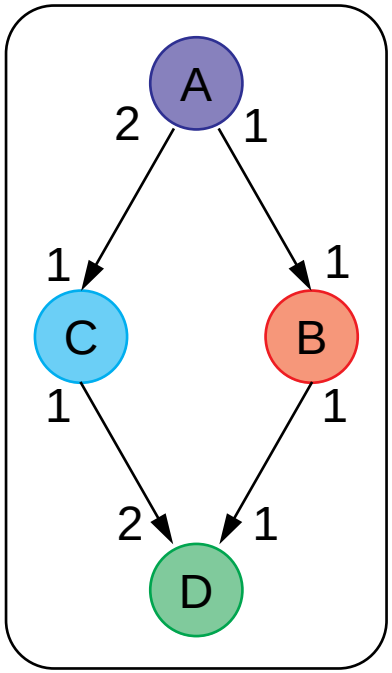


Application designed by Paul Haskell



Multiple models of computation may be used in the same system. Here, dataflow is used for signal processing, while a timed discrete-event system models a communication network.

Synchronous Dataflow (SDF)



$$\begin{bmatrix} 2 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & 1 & 0 & -1 \end{bmatrix} q(V) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$q(V) = \begin{bmatrix} q(A) \\ q(B) \\ q(C) \\ q(D) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Possible Schedules

ABCCD

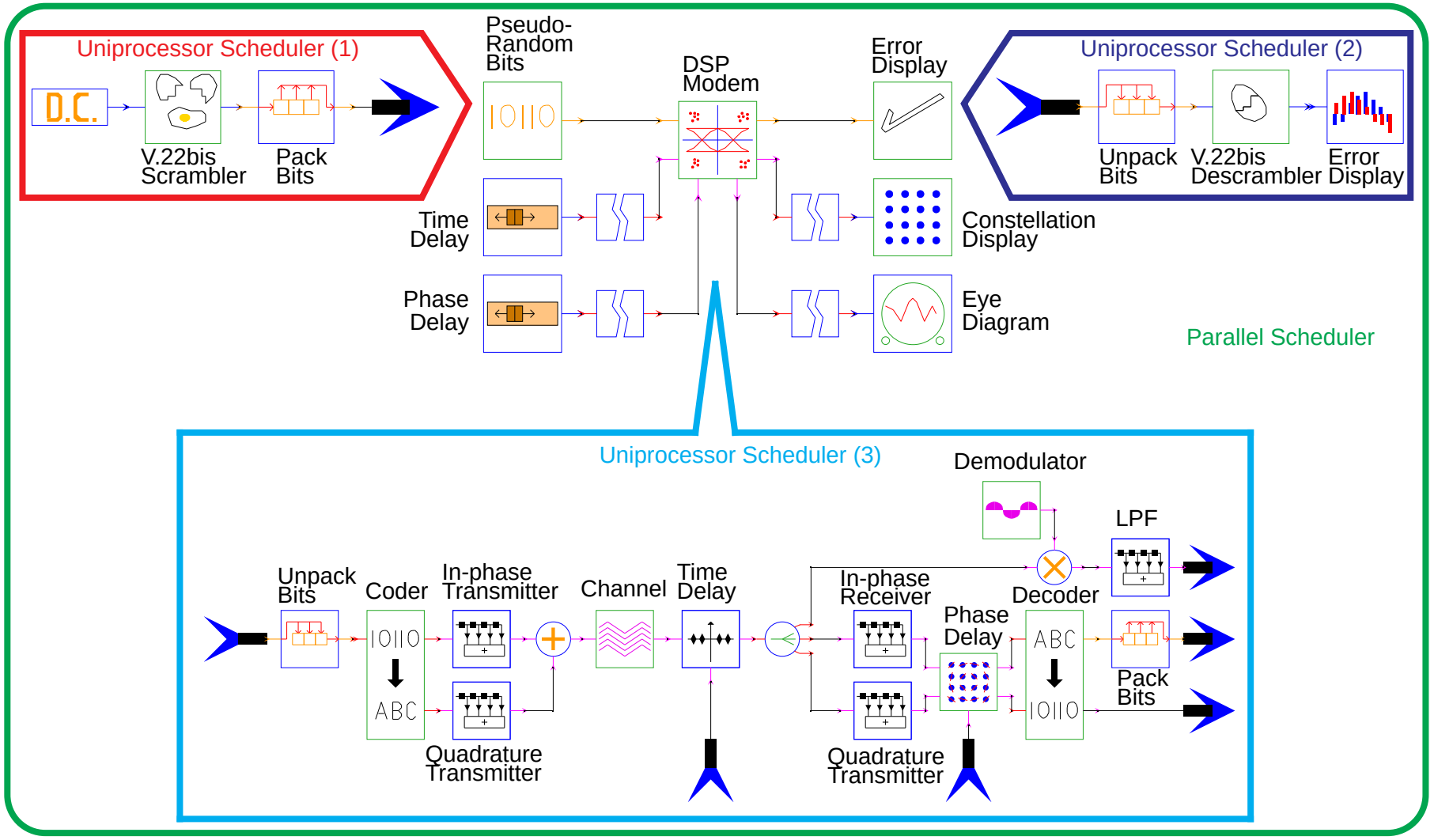
ACBCD

AB(2C)D

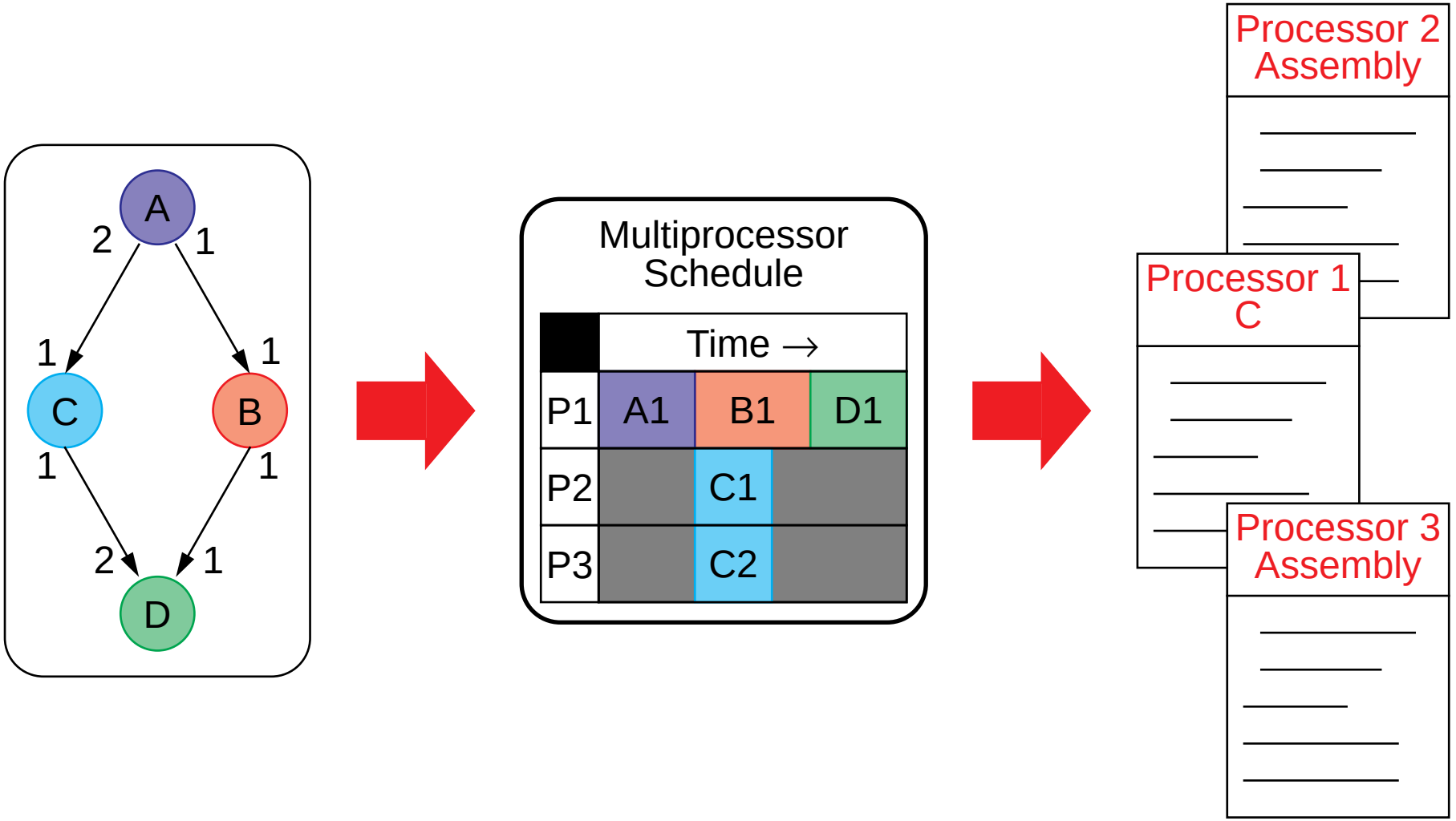
Processor 1

C

Hierarchical Scheduling Framework

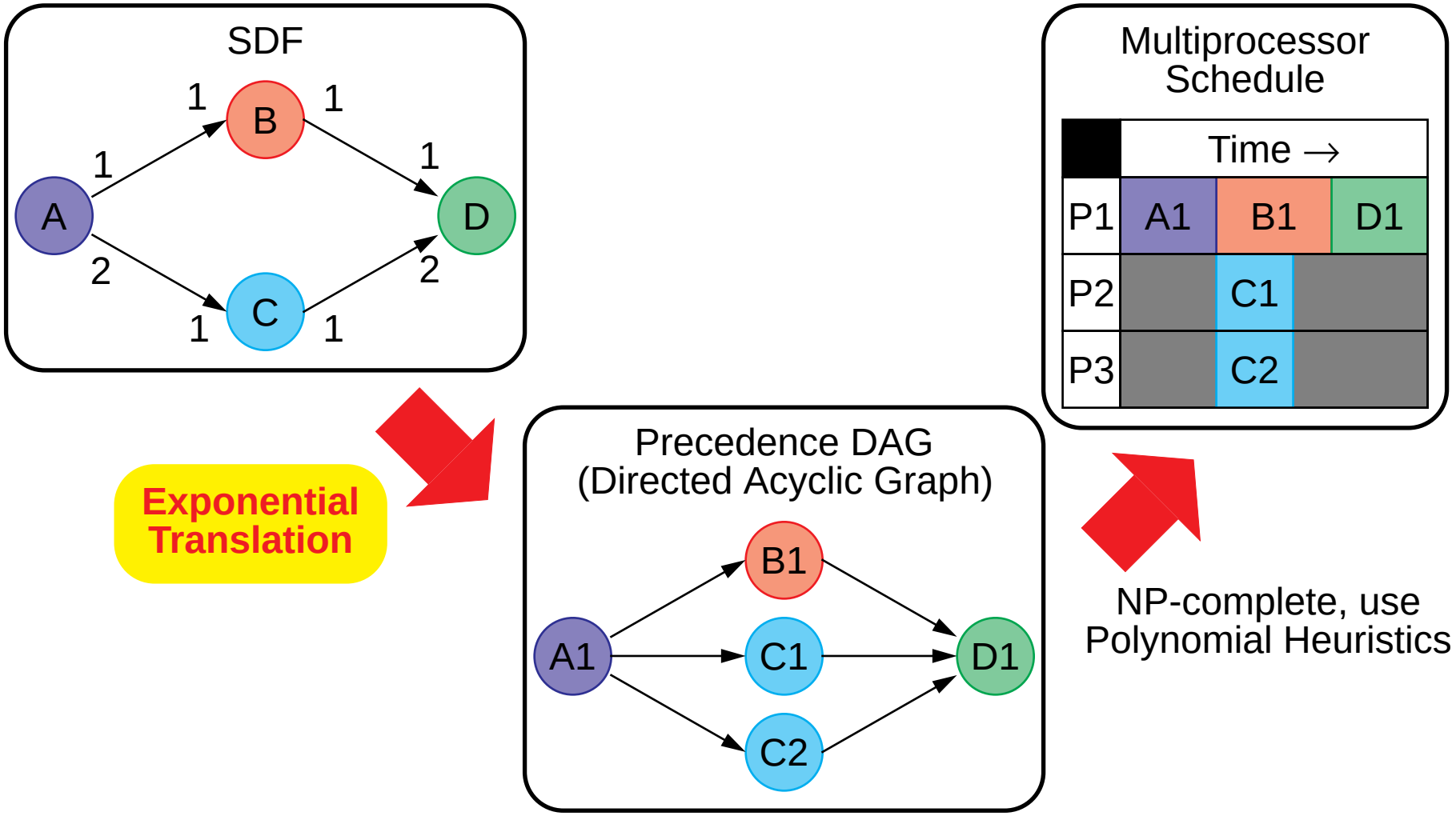


Software Synthesis for Multiple Embedded Processors

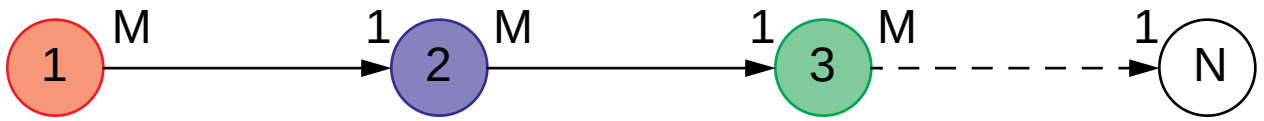


SDF Multiprocessor Scheduling

Past SDF multiprocessor scheduling techniques begin with precedence DAG

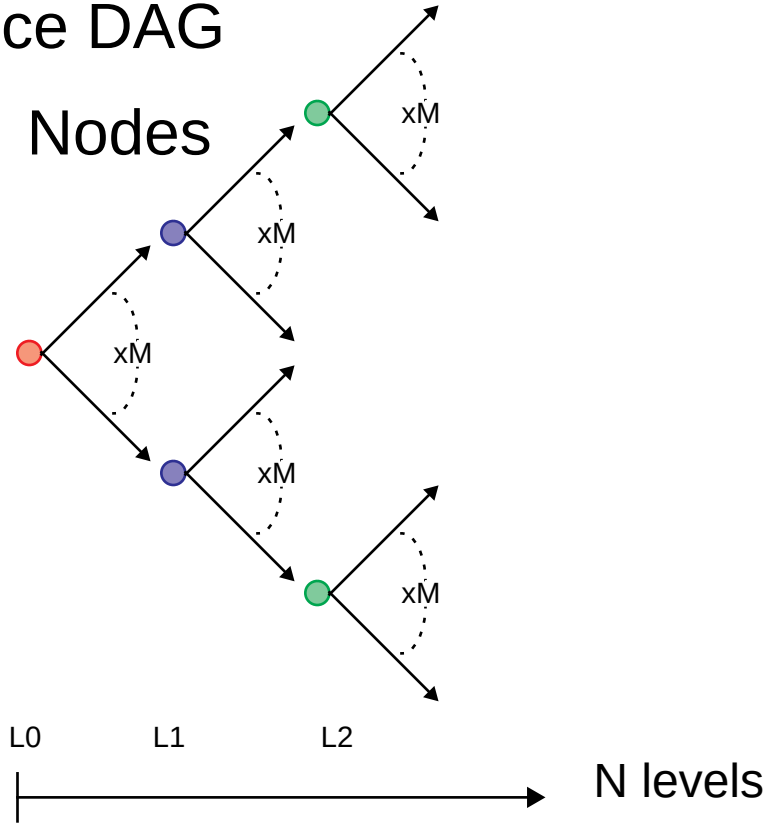


Exponential DAG Expansion Problem



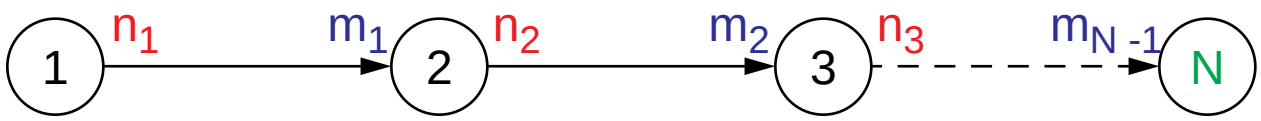
Precedence DAG

$O(M^{N-1})$ Nodes

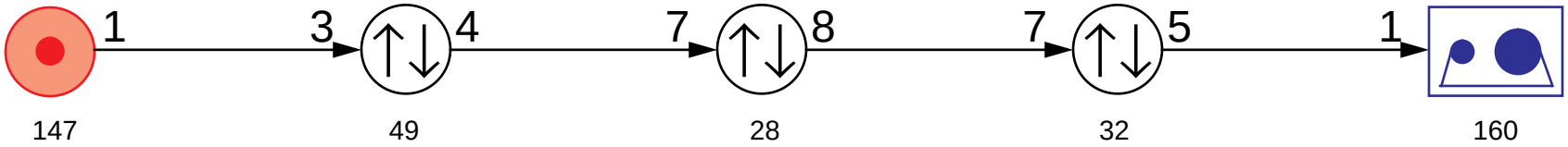


Chains with Mutually Prime Numbers

$$\sum_{j=1}^{N}\left(\prod_{i=1}^{j-1} n_i\right)\left(\prod_{i=j}^{N-1} m_i\right)$$ Precedence DAG Nodes

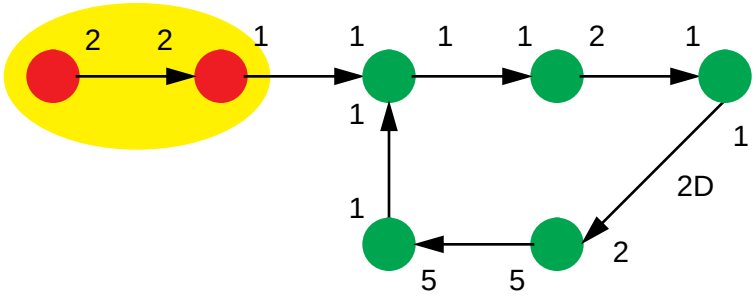


CD to DAT conversion (44.1 kHz to 48 kHz)

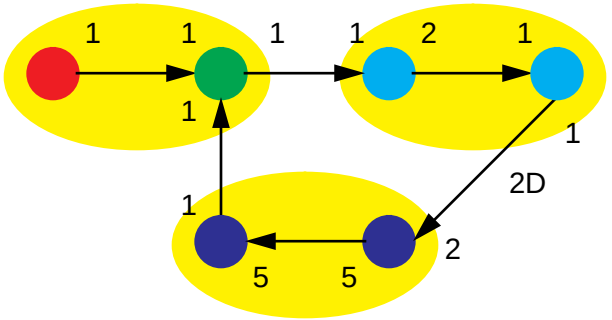


Hierarchical Scheduling Framework

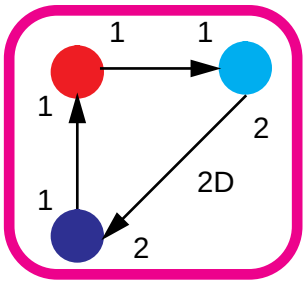
Initialization
Simple clustering



Main Loop
Apply clustering heuristics



Wrapup
Invoke independent schedulers



Multiprocessor Scheduler
Uniprocessor Scheduler 1
Uniprocessor Scheduler 2
Uniprocessor Scheduler 3

Hierarchical Scheduling — Clustering

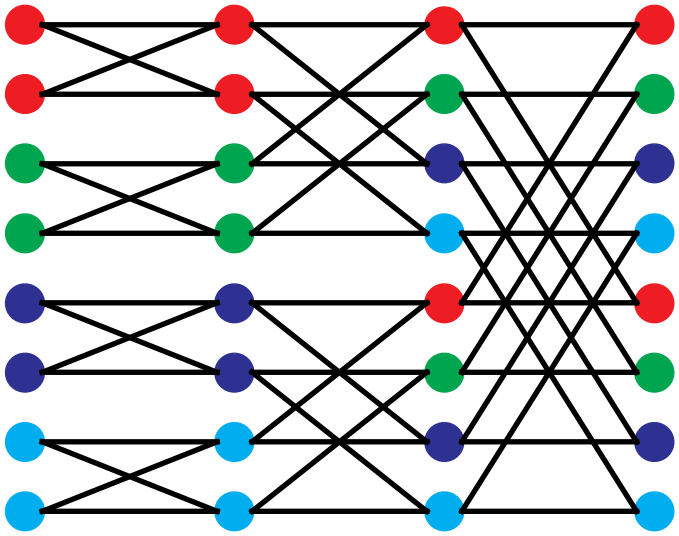
Control growth of nodes by clustering SDF graph

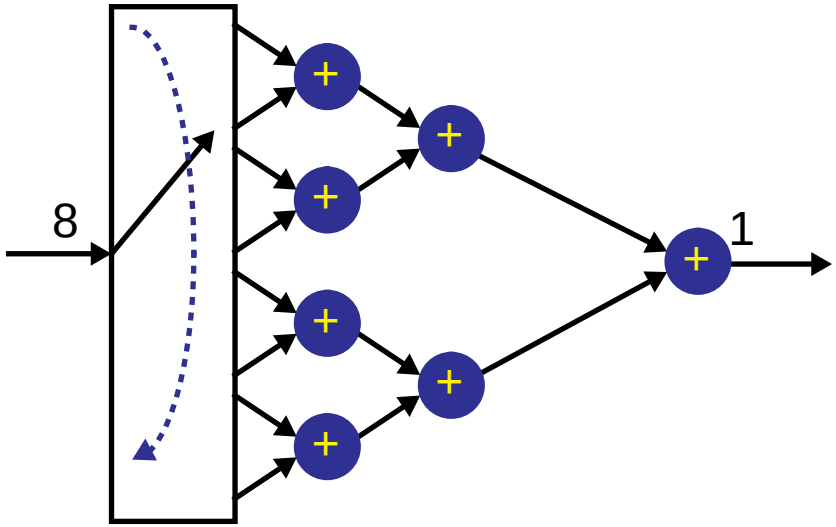
- Simple Heuristics
  - User specified
  - Resource constrained
  - Well-ordered uniform repetition count (URC)
- Interprocessor communication cost (IPC) reduction heuristic

Hierarchical Scheduling — Schedulers

Make use of heterogeneous schedulers:

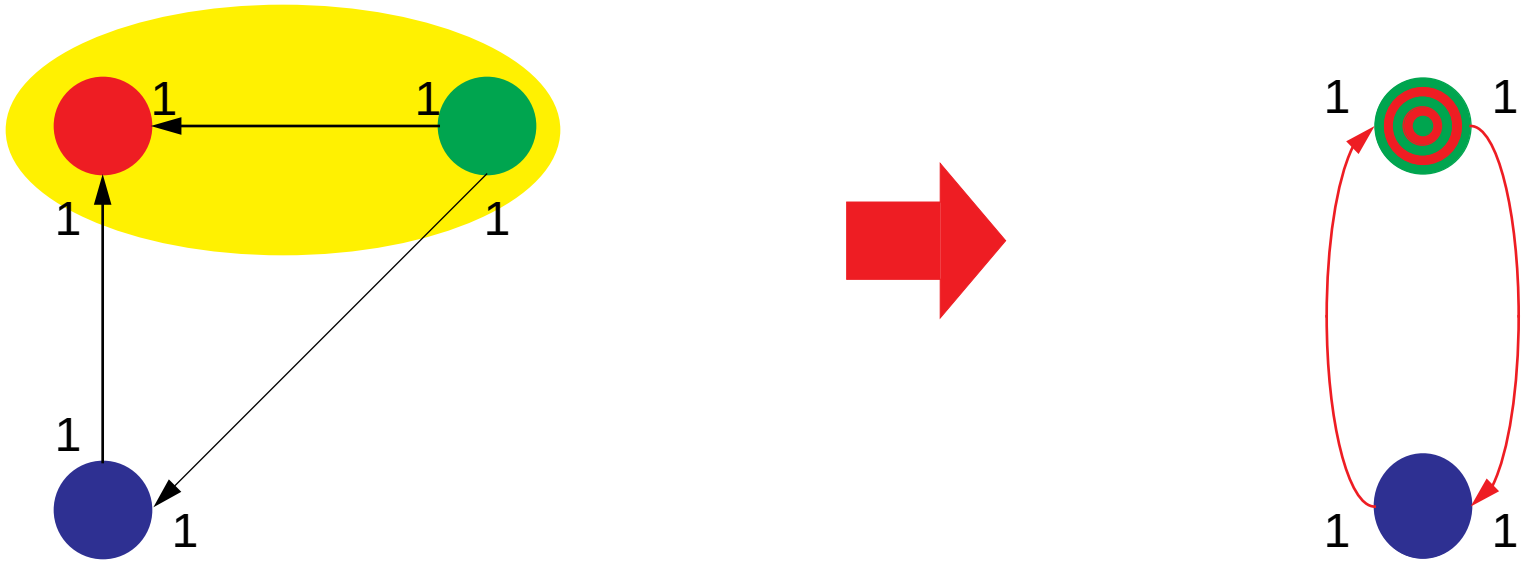
- Uniprocessor schedulers
- Multiprocessor schedulers
- Specialized schedulers





Clustering an SDF Graph

SDF lacks the **composition** property. Thus a cluster of SDF dataflow nodes may change the graph semantics.

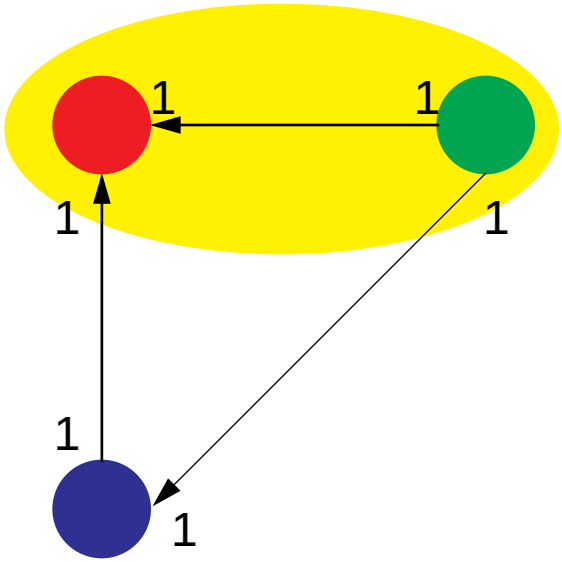


The SDF Composition Theorem

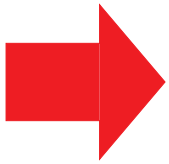
Four sufficient conditions to compose two adjacent SDF nodes into an SDF cluster without changing the semantics of the overall graph eliminate the possibility of:

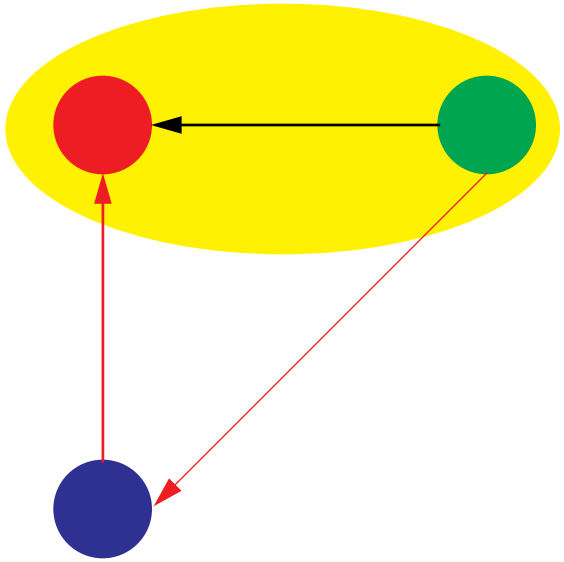
1. Introducing a cycle into the SDF graph
2. Hiding a delay in a cycle
3. Improperly shifting the precedences (1)
4. Improperly shifting the precedences (2)

SDF Composition Theorem
C1: SDF Cycle Introduction



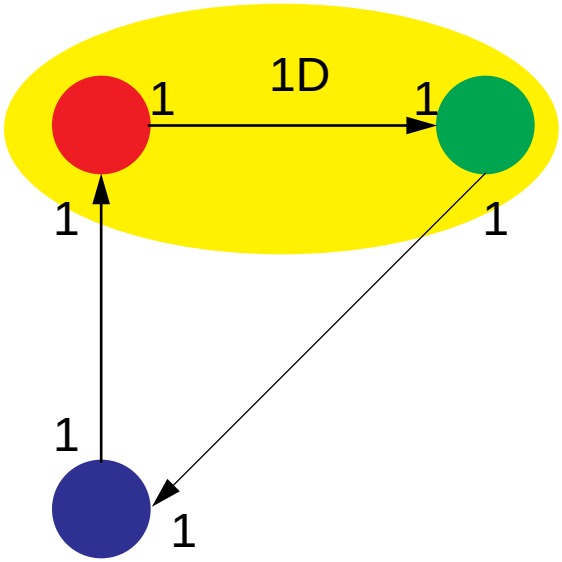
Invalid SDF clustering
which introduces SDF cycle



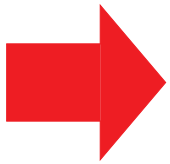


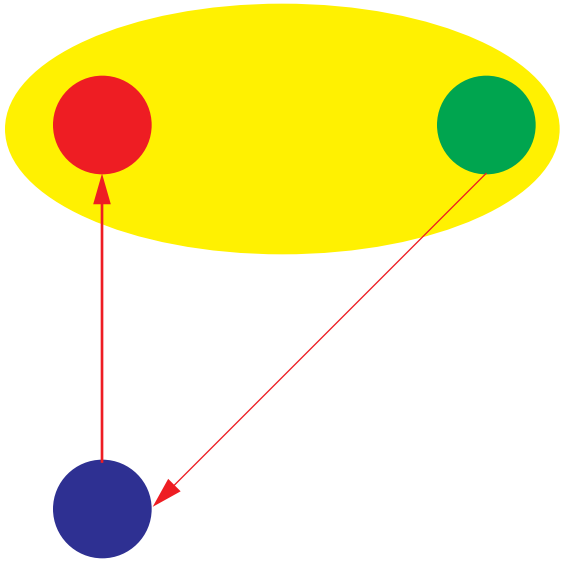
Precedence DAG
showing introduced cycle

SDF Composition Theorem
C2: Hidden Delay in an SDF Cycle



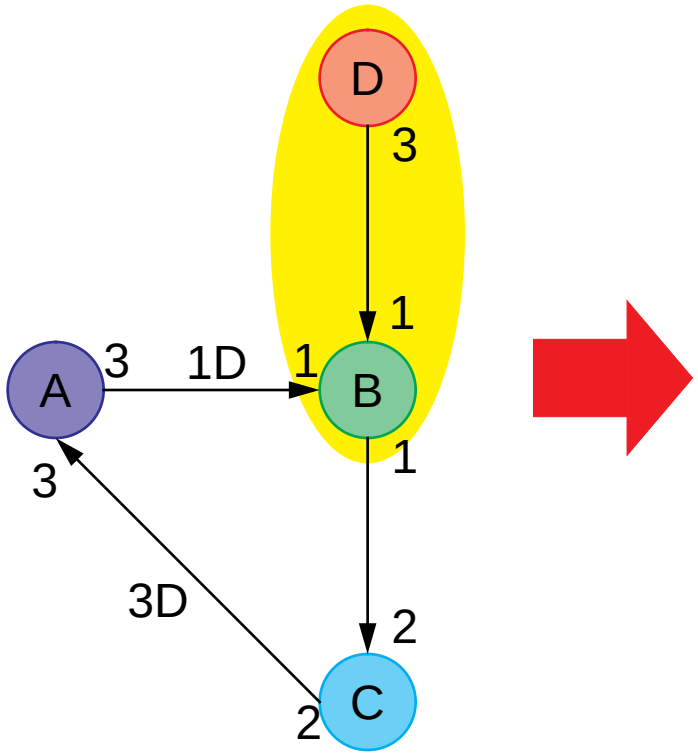
Invalid SDF clustering



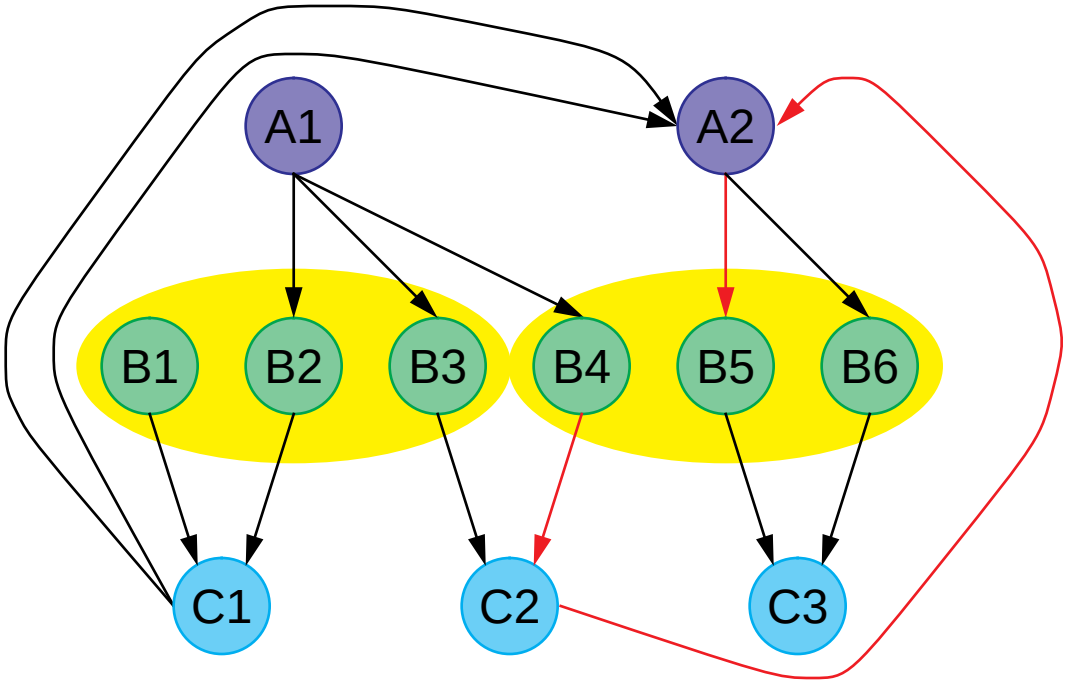


Precedence DAG
showing introduced cycle

SDF Composition Theorem
C3 & C4: Precedence Shift Conditions



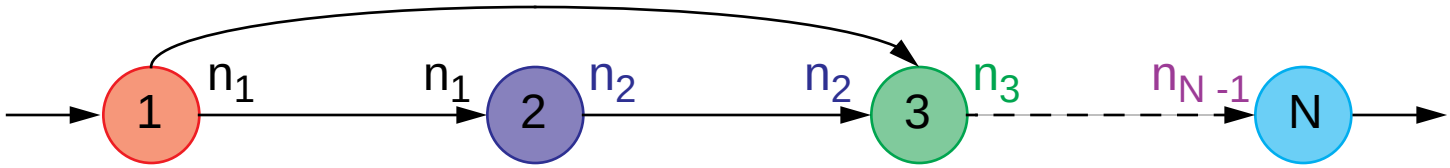
Invalid SDF clustering



Precedence DAG showing introduced cycle

Well-Ordered URC SDF Clustering

N Node URC Dataflow Specification

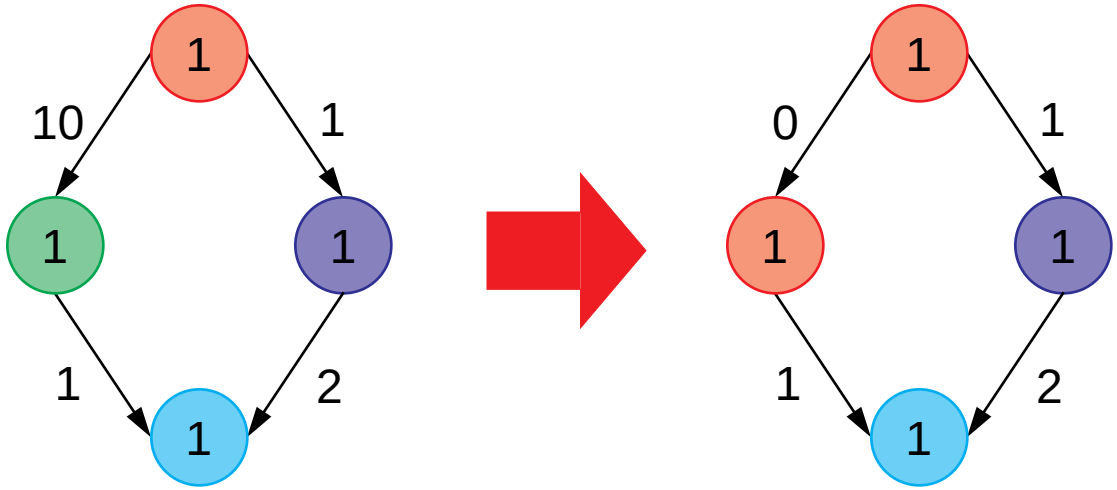


1 Precedence DAG Node

DAG Multiprocessor Scheduling

Common technique for reducing IPC

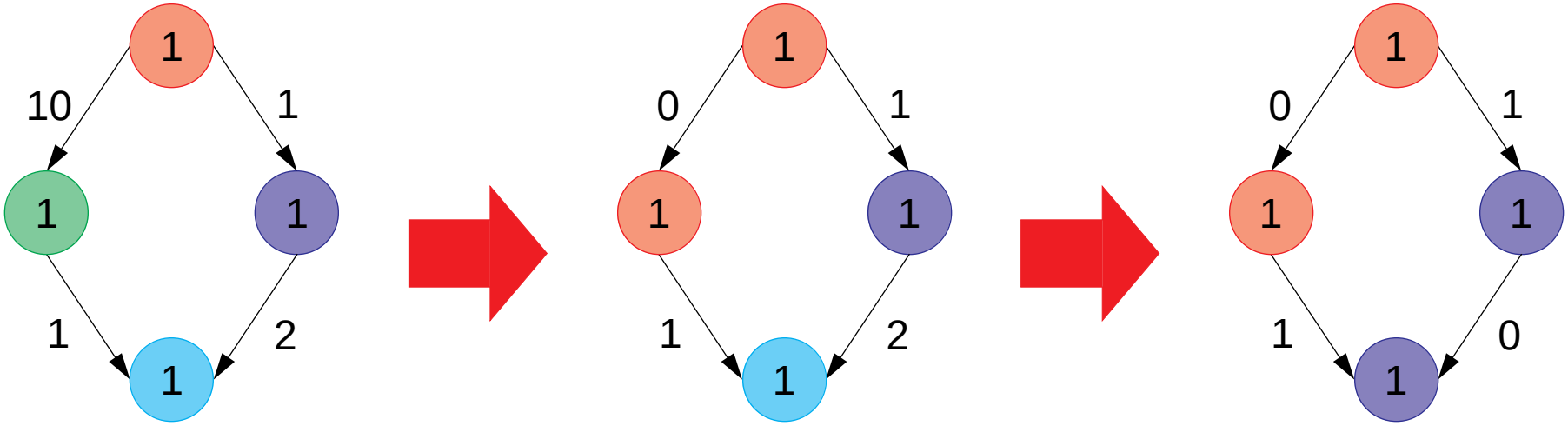
1. Clustering phase – Reduce the **parallel time**



2. Processor assignment phase

Hierarchical Scheduling Main Loop

1. Apply one step of modified Sarkar's multiprocessor clustering heuristic on the **acyclic SDF graph**



2. Test cluster to verify it does not introduce deadlock, and update the repetitions vector and reachability matrix

3. Repeat 1,2 until we reach stopping condition

$$\sum_{v_i' \in V'} q_{G'}(v_i') < K \max(|V'|, P)$$

Performance

System	SDF Graph Size (nodes)	Precedence DAG Size (nodes)	Scheduling Time (seconds)	Parallel Time (clock cycles)	P1: DSP Code Size Assembly (bytes)	P2: Sparc Code Size C (bytes)
FM-Synthesis 128 pt. spectrum	44	14 / 806 57 x smaller	0.47 / 4.35 9.25 x faster	28832 / 28832 no difference	408 / 408 same	34K / 420K 12 x smaller
BPSK (530 bps)	31	9 / 2628 292 x smaller	0.37 / 14.71 40 x faster	41566 / 41368 < 1% difference	424 / 32045 75 x smaller	14K / 56K 4 x smaller
4-QAM (320 bps) eye diagram	59	15 / 9267 618 x smaller	0.91 / 80.87 87 x faster	150123 / 150123 no difference	1421 / 87533 62 x smaller	38K / 63K 1.7 x smaller
4-QAM (640 bps)	52	10 / 3490 349 x smaller	0.69 / 20.1 29 x faster	40037 / 39707 < 1% difference	848 / 29720 35 x smaller	35K / 56K 1.6 x smaller

Conclusions

- Presented **Ptolemy** a heterogeneous system-level design environment for reactive and real-time systems
- Developed a **hierarchical scheduling framework** which uses heterogeneous schedulers to synthesize an application from a high-level dataflow description
- Improved scheduling time and resource usage by **1 - 2 orders of magnitude** using user specified clustering
- Enabled application designers to **synthesize applications previously not possible**