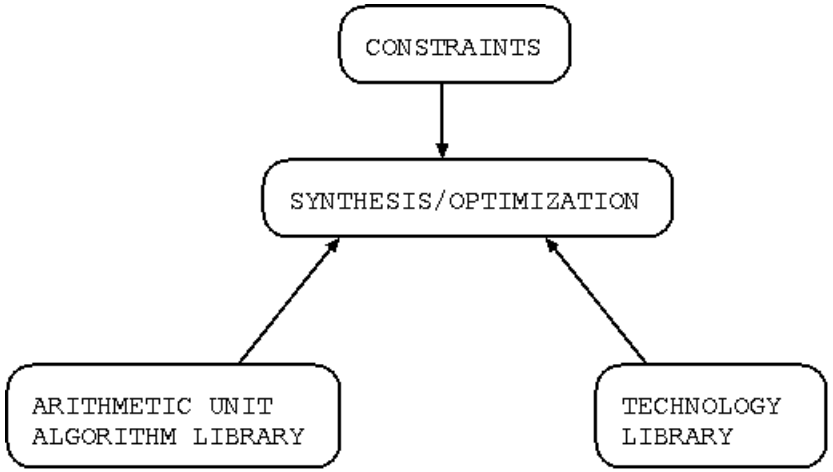
FILTER SYNTHESIS
USING
FINE-GRAIN DATAFLOW GRAPHS

Waqas Akram, Cirrus Logic Inc.

Given a structural filter description, create the most hardware-efficient filter architecture, while satisfying the real-time constraints.

<u>Existing Tools:</u>
HYPER - UC Berkeley
FIR Compiler - Altera
Cadence, Synopsys also have tools

<u>INPUTS:</u> Real-time constraints, Filter description, Module Library



<u>OUTPUT:</u> Optimized Filter Net-list (for direct synthesis into hardware)

SIMPLIFYING ASSUMPTIONS

- **Filter description has no control-flow**
- **Data rate constraints already met**
- **Tool can only perform retiming and folding**
- **Control-flow will be synthesized later**
- **Only tested on small graphs (nodes < 100)**

MODULE LIBRARY

- **Multiple Fine-Grain DFGs for basic functions**
- **Each cell contains resource-usage information**
- **Tied to target technology**

For example, multipliers:

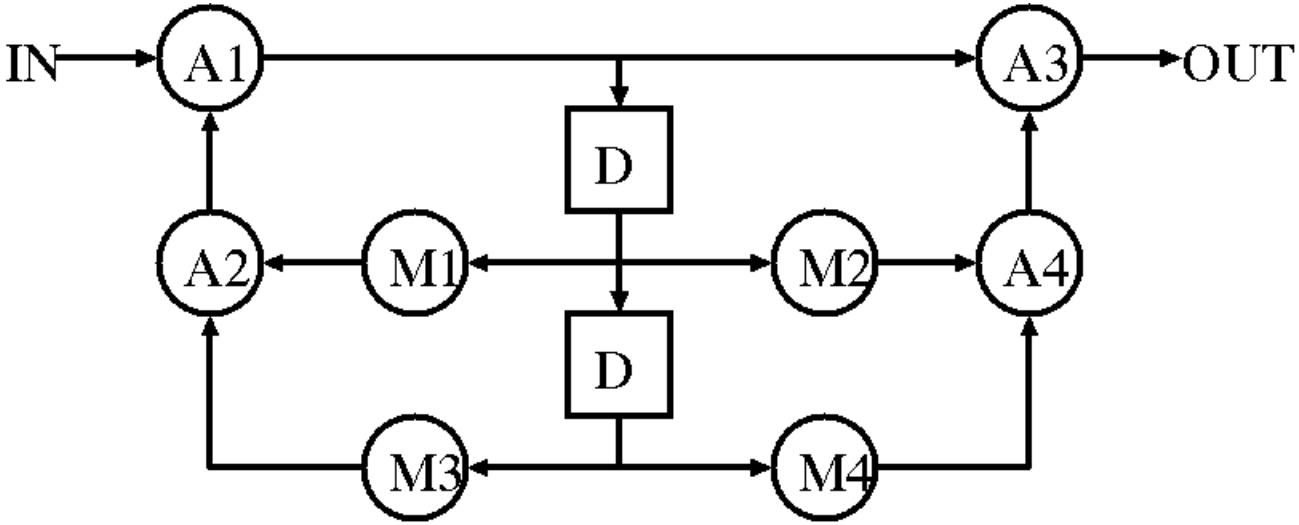
CSA Array,
Shift-and-Add,
Wallace/Dadda Tree,
Booth Recoded

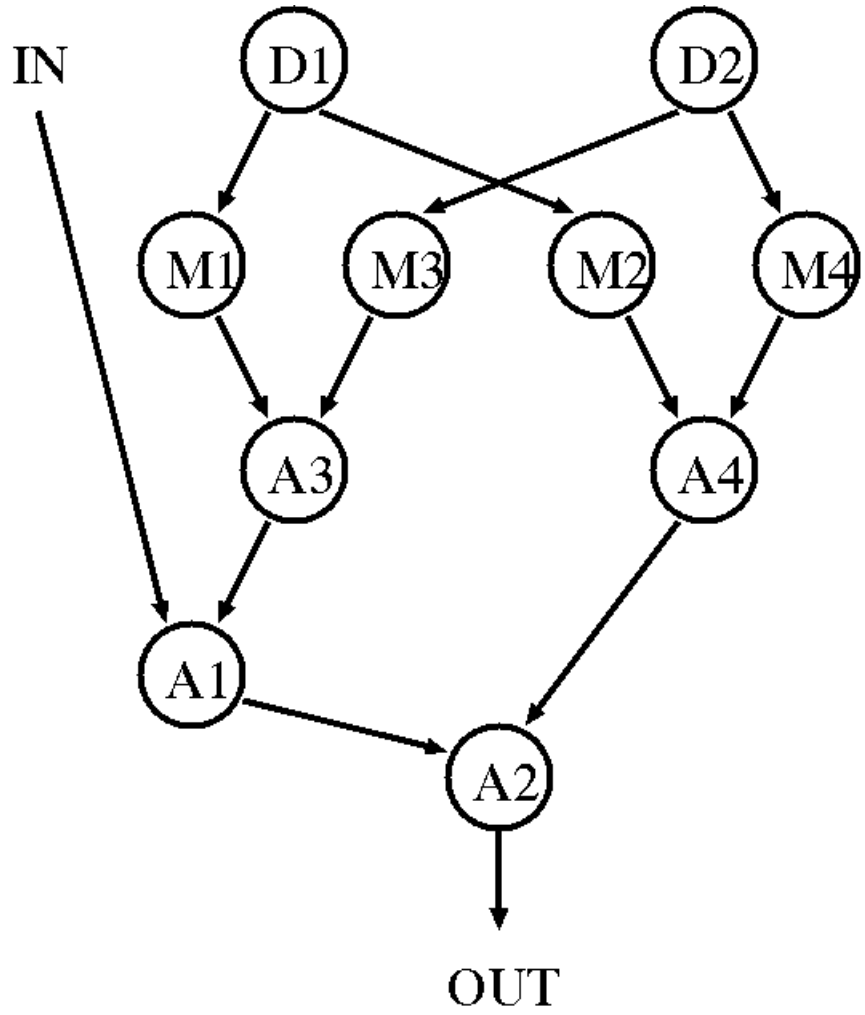
SYNTHESIS STEPS

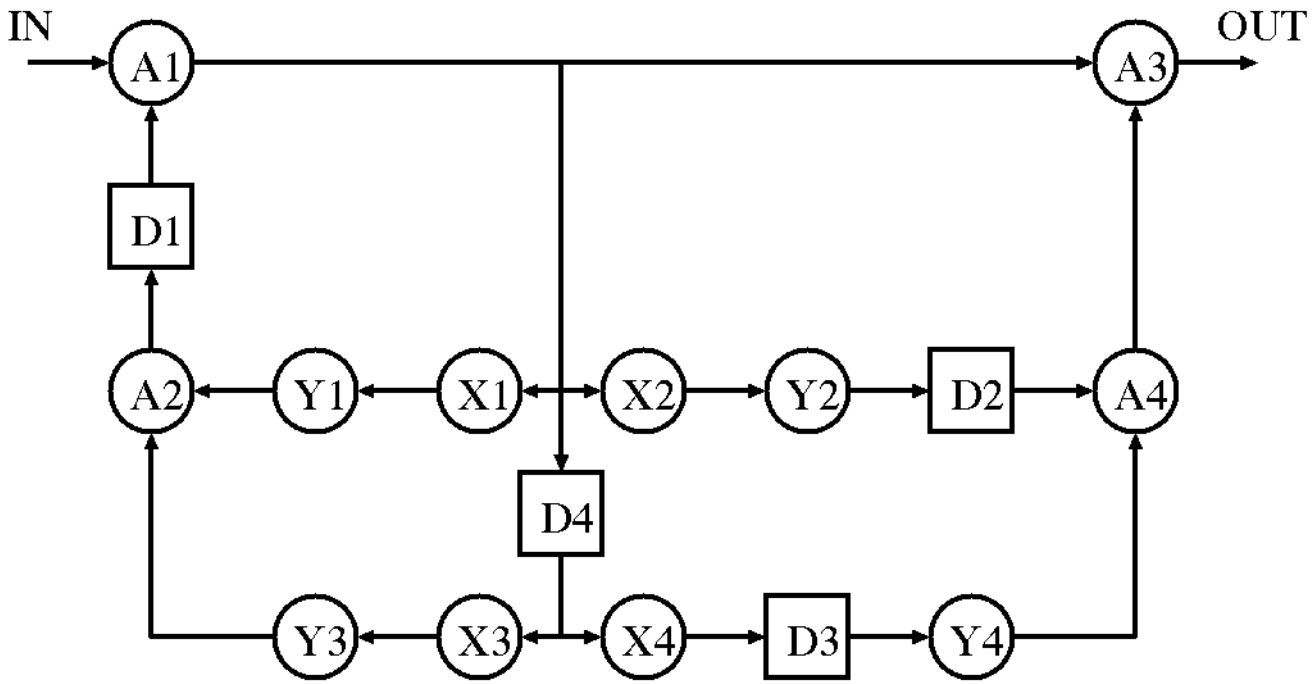
- **Run scheduling algorithm on input DFG**
- **Allocate resources (# of functional units)**
- **Use this allocation as upper-bound**
- **Break all arcs with delays**
- **Create directed acyclic graph**
- **Retime until height reaches iteration bound**
- **Schedule new graph, and compare with bound**
- **Back-track on retiming decision tree, repeat**

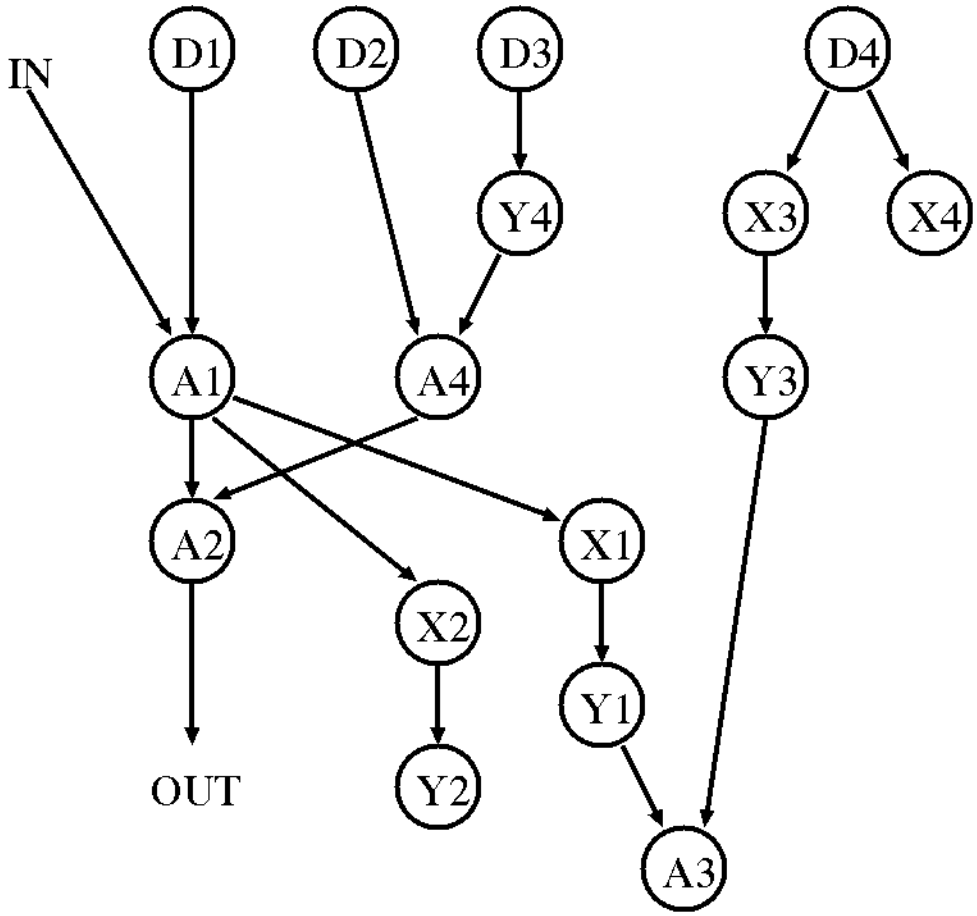
SCHEDULING ALGORITHM

- create priority list for each path in DAG
- longest list becomes critical path
- rank each node according distance from tail
- schedule node(s) with highest rank
- remove node(s) from path(s)
- repeat last 2 steps until all nodes scheduled
- scheduling complexity O(n)


IN
A1
A3
OUT
D
A2
M1
M2
A4
D
M3
M4


IN
D1
D2
M1
M3
M2
M4
A3
A4
A1
A2
OUT


IN
A1
A3
OUT
D1
A2
Y1
X1
X2
Y2
D2
A4
D4
Y3
X3
X4
D3
Y4


IN
D1
D2
D3
D4
Y4
X3
X4
A1
A4
Y3
A2
X1
X2
Y1
OUT
Y2
A3

RESULTS/COMPARISON

BIQUAD/HYPER					BIQUAD/FINE-GRAIN			
TIME UNIT	ADD	MULT 1	MULT 2		TIME UNIT	ADD 1	ADD 2	PART. MULT
1	A2	M3	M4		1	A1	Y4	X3
2	A1				2	A4	Y3	X1
3	A4	M1	M2		3	A3	Y1	X2
4	A3				4	A2	Y2	X4
1 Adder, 2 Multipliers, 5 Registers					1 Adder, 1 (effective) Multiplier, 5 Registers			
Assumptions: 1 Adder = 1 TU, I Multiply = 2 TU					Assumptions: 1 Adder = 1 TU, 1 Partial Multiply = 1 TU			

CONCLUSIONS

- **Successful implementation of synthesis**
- **Subset of transformations (no unfolding)**
- **Constrictive input conditions**
- **Produces DFG with control block hierarchy**
- **Control blocks are simply firing schedules**
- **Still need to synthesize control logic**