Algorithm to architecture transformation

Dr. Shubhajit Roy Chowdhury,
Centre for VLSI and Embedded Systems Technology,
IIIT Hyderabad

Email: src.vlsi@iiit.ac.in

The process

- Input: Vague ideas about the system to be developed
- Output: An architectural design of the system
- Methodology:

Step 1: The vague ideas about the system have to be formalized in terms of algorithms. The algorithm depicts the behavior of the system

Step2: The algorithm has to be transformed into an architectural design

High-level synthesis

- Sequential operation is not the most abstract description of behavior.
- We can describe behavior without assigning operations to particular clock cycles.
- High-level synthesis (behavioral synthesis) transforms an unscheduled behavior into a register-transfer behavior.

Specifications needed for high level synthesis

- **Resources**
- Functional resources

-Primitive resources

-Application specific resources

- Memory resources
- Interface resources

- **Constraints**

-Interface constraints

-Implementation constraints

Tasks in high-level synthesis

- Scheduling: determines clock cycle on which each operation will occur.
- Binding (allocation): chooses which function units will execute which operations.

Functional modeling code in VHDL

```
o1 <= i1 or i2;
if i3 = ‘0’ then
     o1 <= ‘1’;
     o2 <= a + b;
else
     o1 <= ‘0’;
end if;
```

clock cycle boundary can be moved to design different register transfers

Data dependencies

- Data dependencies describe relationships between operations:
  - x <= a + b; value of x depends on a, b
- High-level synthesis must preserve data dependencies.

Data flow graph

- Data flow graph (DFG) models data dependencies.
- Does not require that operations be performed in a particular order.
- Models operations in a basic block of a functional model
- Requires single-assignment form.

Data flow graph construction

original code:

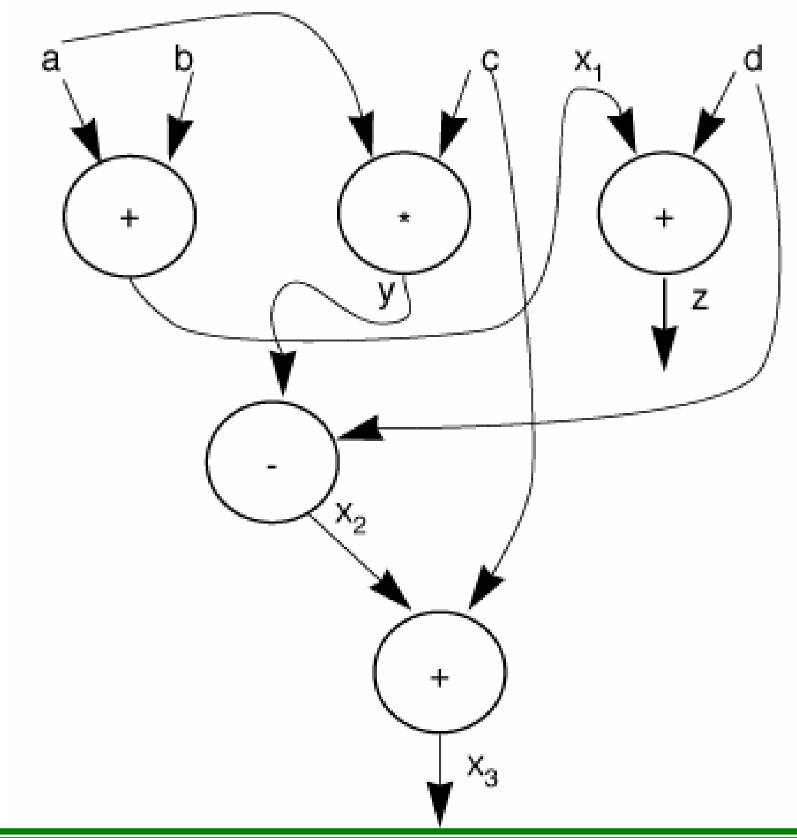
```
x <= a + b;
y <= a * c;
z <= x + d;
x <= y - d;
x <= x + c;
```

single-assignment form:

```
x1 <= a + b;
y <= a * c;
z <= x1 + d;
x2 <= y - d;
x3 <= x2 + c;
```

Data flow graph construction, cont'd

Data flow forms directed acyclic graph (DAG):

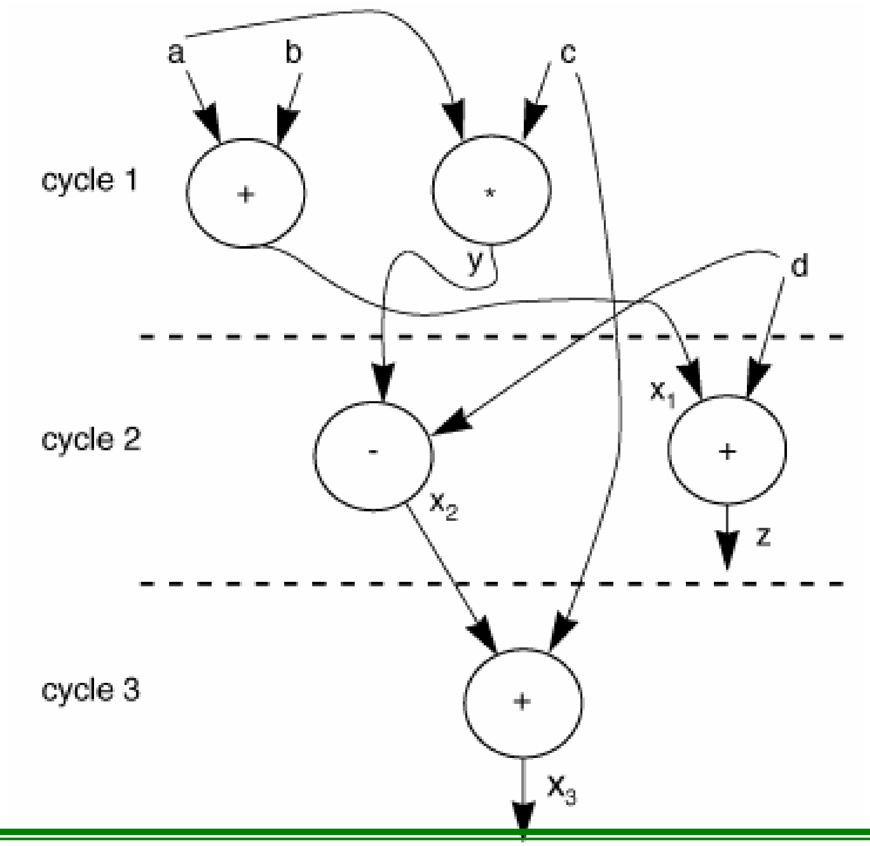


Goals of scheduling and allocation

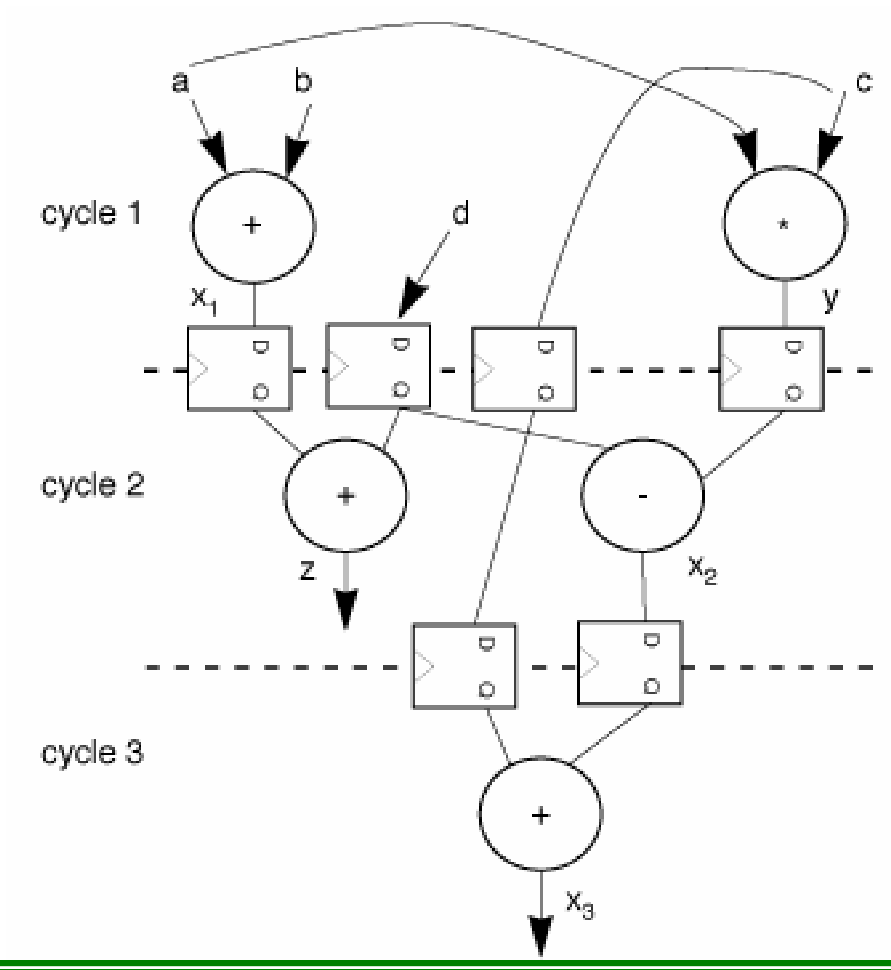
- Preserve behavior—at end of execution, should have received all outputs, be in proper state (ignoring exact times of events).
- Utilize hardware efficiently.
- Obtain acceptable performance.

Data flow to data path-controller

One feasible schedule for last DFG:



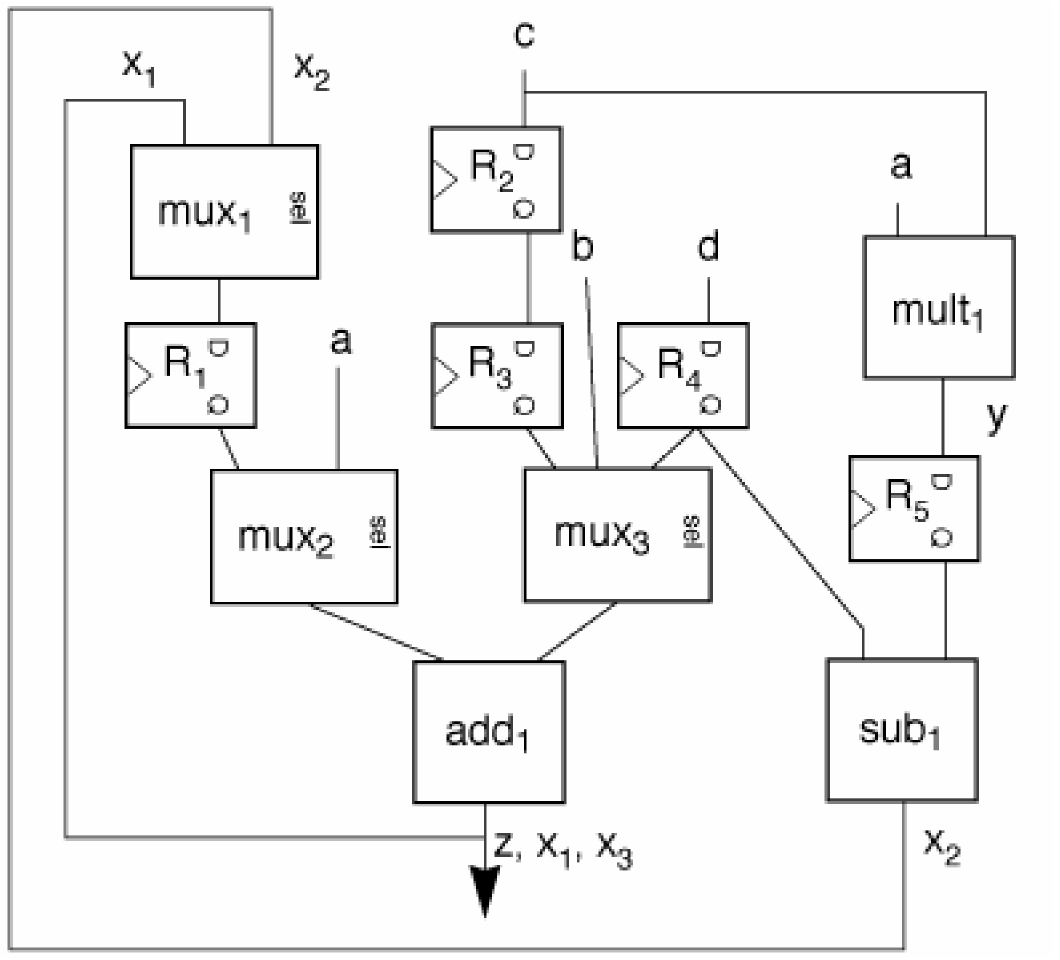
Binding values to registers



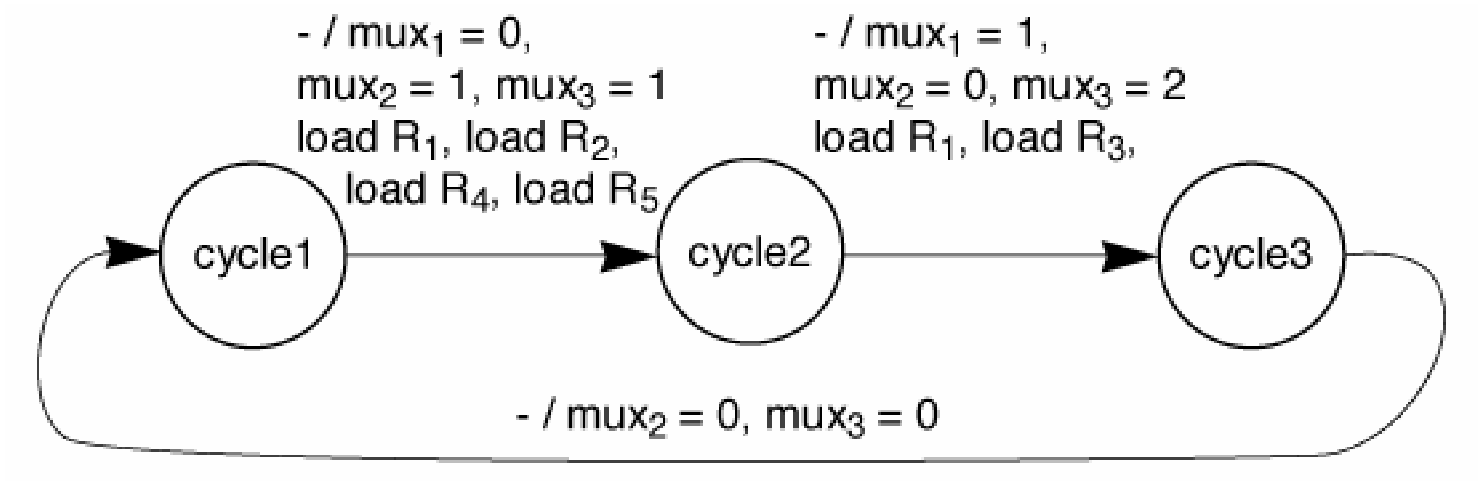
registers fall on clock cycle boundaries

Choosing function units

muxes allow function units to be shared for several operations



Building the sequencer



sequencer requires three states,
even with no conditionals

Choices during high-level synthesis

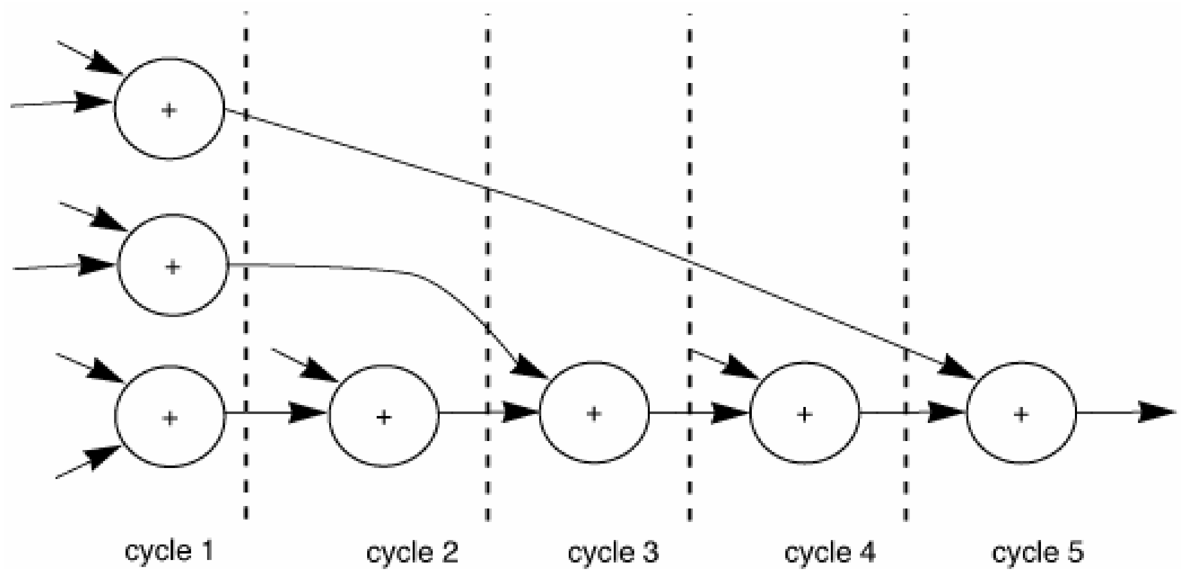
- Scheduling determines number of clock cycles required; binding determines area, cycle time.
- Area tradeoffs must consider shared function units vs. multiplexers, control.
- Delay tradeoffs must consider cycle time vs. number of cycles.

Finding schedules

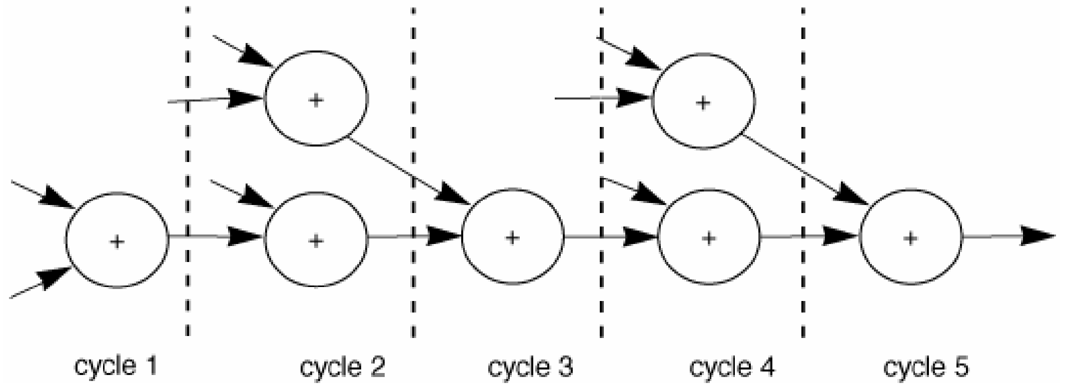
- Two simple schedules:
  - As-soon-as-possible (ASAP) schedule puts every operation as early in time as possible.
  - As-late-as-possible (ALAP) schedule puts every operation as late in schedule as possible.
- Many schedules exist between ALAP and ASAP extremes.

ASAP and ALAP schedules

ASAP

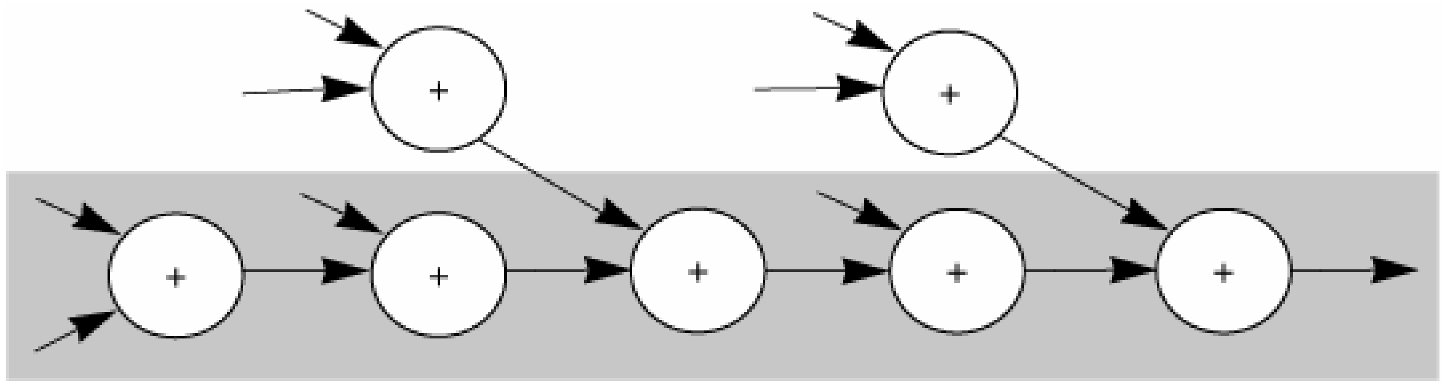


ALAP



Critical path of schedule

Longest path through data flow determines minimum schedule length:



Questions?