Data Flow Graphs

Intro

Sources: Gang Quan
Computational Models

- What:
  - A conceptual notion for expressing the function of a system
    - E.g. DFG, FSM, Petri net, Turing machine, etc.
- Computational Models & Languages
  - Models express the behavior, languages capture models
  - Models are conceptual, languages are concrete
- What is in a computational model
  - A set of objects
  - Rules
  - Semantics
Data Flow Graph (DFG)

- A modem communications system

- Each box is a single function or sub systems
- The activity of each block in the chain depends on the input of the previous block
- Data driven
  - Each functional block may have to wait until it receives a "certain amount" of information before it begins processing
  - Some place to output the results
Data Flow Graph

- **Definition**
  - A directed graph that shows the **data dependencies** between a number of functions
  - $G=(V,E)$
    - Nodes (V): each node having input/output data ports
    - Arces (E): connections between the output ports and input ports
  - **Semantics**
    - Fire when input data are ready
    - Consume data from input ports and produce data to its output ports
    - There may be many nodes that are ready to fire at a given time
Data Flow Graph Construction

\[ x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]

\[ x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \]
\[ x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \]

\[ x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \]
Data flow graph construction

original code:
\[ x \leq a + b; \]
\[ y \leq a \times c; \]
\[ z \leq x + d; \]
\[ x \leq y - d; \]
\[ x \leq x + c; \]
Data flow graph construction

original code:

\[ x \leq a + b; \]
\[ y \leq a \times c; \]
\[ z \leq x + d; \]
\[ x \leq y - d; \]
\[ x \leq x + c; \]

single-assignment form:

\[ x_1 \leq a + b; \]
\[ y \leq a \times c; \]
\[ z \leq x_1 + d; \]
\[ x_2 \leq y - d; \]
\[ x_3 \leq x_2 + c; \]
Data flow graph construction

single-assignment form:

\[
\begin{align*}
    x_1 & \leq a + b; \\
    y & \leq a \times c; \\
    z & \leq x_1 + d; \\
    x_2 & \leq y - d; \\
    x_3 & \leq x_2 + c;
\end{align*}
\]
Design Issues

- Allocating
- Mapping
- Schedule
- Memory management
- Construction and usage of the queues
Goals

- Guarantee correct behavior
- Utilize hardware efficiently.
- Obtain acceptable performance.
Decide the numbers and types of different functional units
  - E.g. register allocation

\[
\begin{align*}
\text{…..} \\
x &= a + b; \\
y &= a + c; \\
x &= x - c; \\
\text{…..}
\end{align*}
\]

three registers

\[
\begin{align*}
\text{…..} \\
x &= a + b; \\
y &= a + c; \\
x &= x - c; \\
\text{…..}
\end{align*}
\]

\[
\begin{align*}
\text{…..x …} \\
\text{…..y…..}
\end{align*}
\]
Mapping

- Distributing nodes to different functional units on which they will fire
  - Functional units may provide different functions
    - Adder or ALU, MUX or buses, etc
  - Functional units may have different delay
    - Ripple adder or look ahead adder
  - Determines area, cycle time.
A Mapping Example

Subject to:

- Two adders
- Four registers
- b and e cannot be assigned to the same register
A Mapping Example

Subject to:
- Two adders
- Three registers
- a and e cannot be assigned to the same register

Mapping may not be unique!
Scheduling of DFG

- **Schedule**
  - Creating the sequence in which nodes fire
  - Determines number of clock cycles required

- **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
Nodes fire whenever the input data are available.
Nodes fire when absolutely necessary.
More about ASAP and ALAP

- Unlimited resources
  - No limit for the number of registers, adders, etc
- Longest path through data flow determines minimum schedule length
- Mobility
  - $t_L - t_S$
The node mobility represents its flexibility in the fire sequence.
Restrained Scheduling

- Time constraints
  - Time is given, minimize the resource
- Resource constraints
- NP problem
Time Constraints

\[ T \begin{bmatrix} 6 & 7 & 8 \\ +/ & 2 & 1 & 1 \\ */ & 2 & 2 & 1 \\ ** & 1 & 1 & 1 \\ sqrt & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \]
Resource Constraints

- Resource is given, minimize the long time
- List based scheduling
  - Maintain a priority based ready list
    - The priority can be decide by mobility for example
  - Fire the nodes according to their priorities until all the resource are used in that stage
List Based Scheduling

S.t: one +/-, one *//
List Based Scheduling

- A general ASAP
- Priority based ready list
x <= a + b;
if ( x > 100)
  y <= a * c;
else
  y <= a + c;
endif
Control/Data Flow Graph

Definition
- A directed graph that represents the control dependencies among the functions
  - branch
  - fall-through
- $G=(V,E)$
  - Nodes ($V$)
    - Encapsulated DFG
    - Decision
  - Arces ($E$)
    - flow of the controls
fun0();
if (cond1) fun1();
else fun2();
fun3();
switch(test1) {
case 1: fun4();
    break;
case 2: fun5();
    break;
case 3: fun6();
    break;
}
fun7();
fun0();
while(cond1) {
    fun1();
}
fun2();
Design Issues

- Code optimization
  - Loop optimization, dead code detection
- Register allocation
Summary

- **Data Flow Graph (DFG)**
  - models data dependencies.
  - Does not require that nodes be fired in a particular order.
  - Models operations in the functional model—no conditionals.
  - Allocation and Mapping
  - Scheduling – ASAP, ALAP, List-based scheduling

- **Control/Data Flow Graph**
  - Represents control dependencies