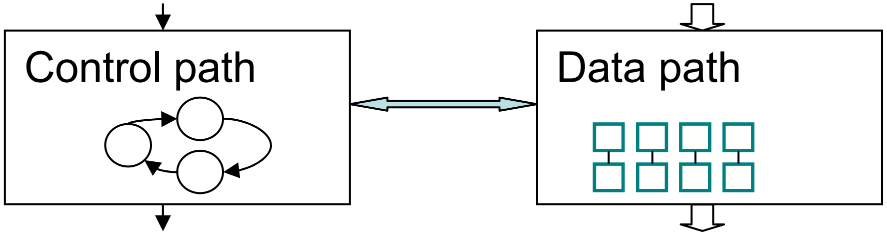
ADL transformation using Process Graph and DFA

TID 2007

Tomáš Novotný

Opening

- ADL (Architecture definition language)
  - Description of architecture resources and behavior
  - High abstraction of description
  - We need transformation to HDL (Hardware description language)
  - HDL design of architecture decomposed into two parts



- Three main components have to be generated
  1. Data path
     - Process graph representation
  2. Decoder
     - Description of DFA
  3. Control path
     - FSM representation

Process Graph

Definition

Process graph is labeled multidigraph

$G = (A_v, A_E, V, E, \varepsilon, l_V, l_E)$ where :

- V is a finite nonempty set of vertexes.
- E is a finite set of edges.
- A_v, A_E are finite sets of the available vertex and edge labels; $A_v \cap A_E = \emptyset$
- ε is a function $\varepsilon : E \rightarrow V \times V \setminus \{(x,x) \mid x \in V\}$, we call it incidence function.
- $l_V : V \rightarrow A_v$ and $l_E : E \rightarrow A_E$ are two function (describe the labeling of vertexes and edges).

Process Graph

- Represents architecture data path.
- Vertex represents building block of architecture, which has sequential behavior as the VHDL process.
- Edges represents block's interconnection (control and data signals).

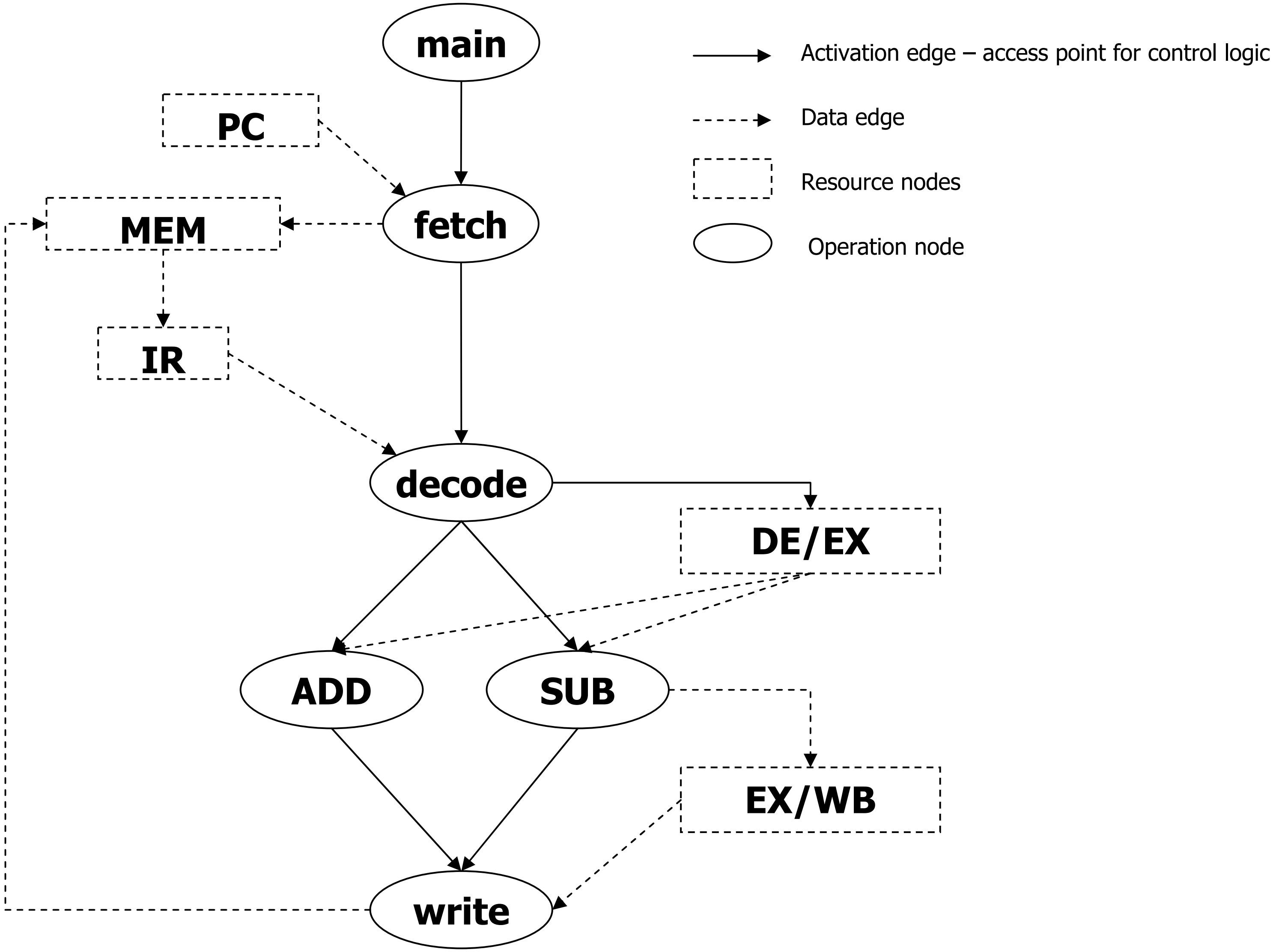
$V = V_{store} \cup V_{proc}$, where:

- V_{store} represents vertexes of ISAC resources.
- V_{proc} represents vertexes of ISAC operations.

$E = E_{act} \cup E_{data}$, where:

- V_{act} represents activation signals.
- V_{data} represents data signals.

Process Graph - Example

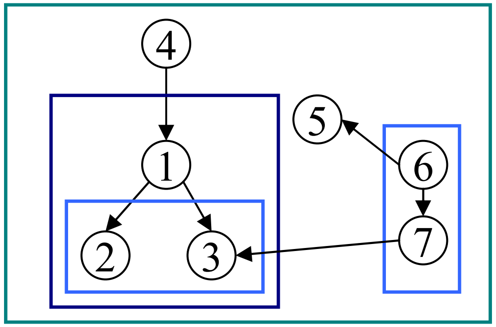


Clustered Graph

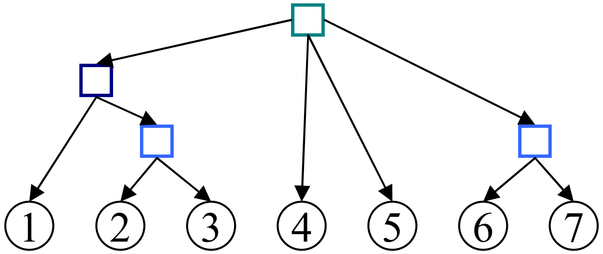
Definition

- A Clustered graph C = (G, T) consists of a directed graph G = (V,E) and rooted tree T = (V_T,E_T), such that the leaves of T are exactly the nodes of G.
- Each non-leave vertex $v \in V_T$ represents a cluster V(*v*), the subset of the vertices of G that are leaves of the subtree rooted at *v*.

- Clusters: V(*v*)
  - Contain vertices
  - Or other cluster
- Rooted tree: T
  - No cluster overlap
  - Single root cluster
- Graph: G
  - Edges connect vertices but not clusters



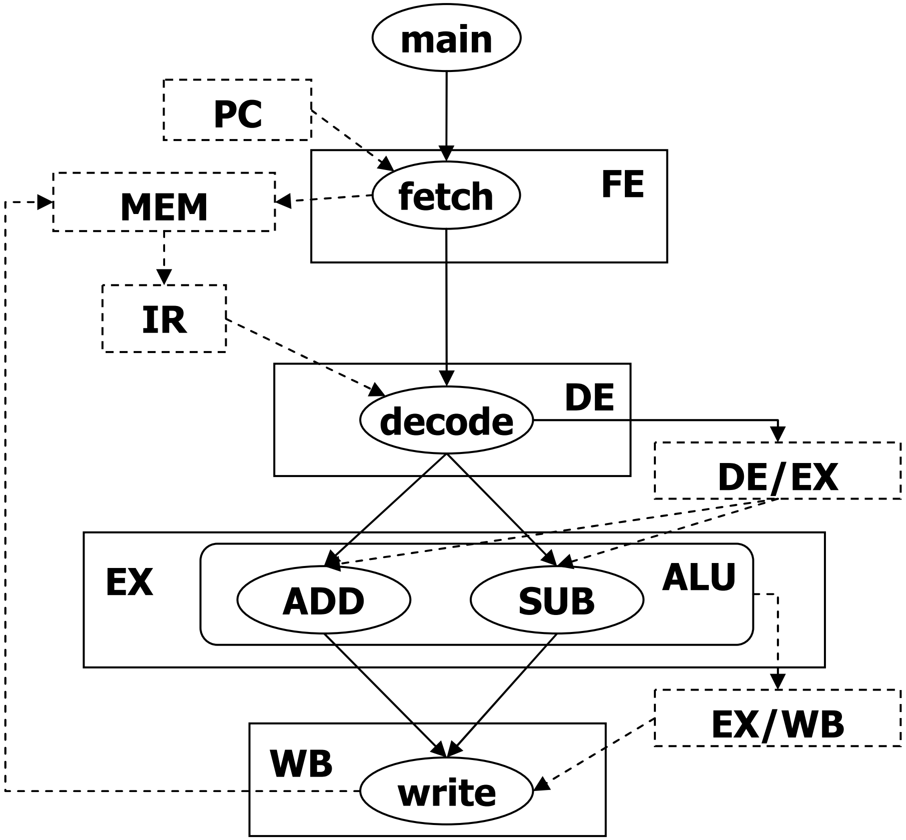
G

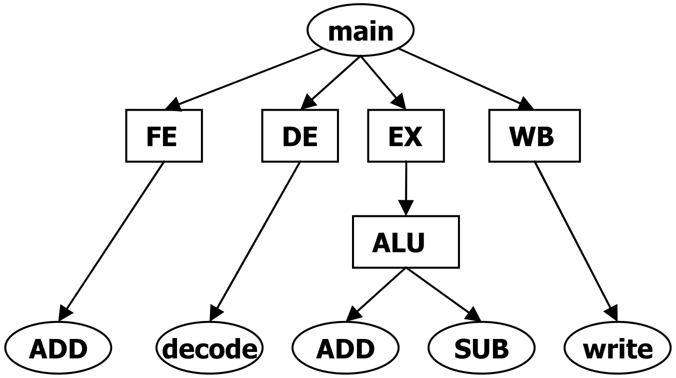


T

Process Graph - Hierarchy

- Operations are assigned into functional units (FU)
- FU or operations are assigned into pipeline stages



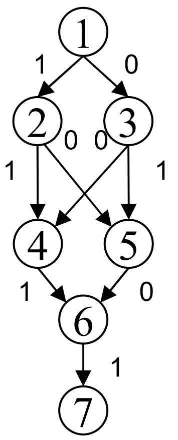


Transformation with Process Graph

- Process graph is constructed directly from ADL.
- Information from Process graph used for generating of VHDL - RTL level .
- For each vertex $v \in V_{store}$ VHDL entity is generated with the name from $l_{V.}$
- For each vertex $v \in V_{store}$ VHDL process is generated with the name from $l_{V.}$
- From edges VHDL signals are generated. Some interconnection is generated.
- Hierarchy
  - For each cluster V(v) VHDL entity is generated.
  - Processes are components of those entities.
- Controller and decoder entity generated from different models.

Decoder

- DFA automata
- Generation of description of DFA in VHDL
- We have oriented graph (DFA) and we need to know all possible paths.
- Two ways of VHDL description



1.

```
case reginstreg(1) is
  when ‘1’ =>
    case reginstreg(2) is
      when ‘1’ =>
      when ‘0’ =>
    end case;
  when ‘0’ =>
    case reginstreg(2) is
      when ‘1’ =>
      when ‘0’ =>
    end case;
end case;
```

OR

2.

```
if (v_reginstreg_1 = “1111”)
  -- do something
end if;
if (v_reginstreg_2 = “1001”)
  -- do something
end if;
if (v_reginstreg_3 = “0011”)
  -- do something
end if;
if (v_reginstreg_4 = “0101”)
  -- do something
end if;
```

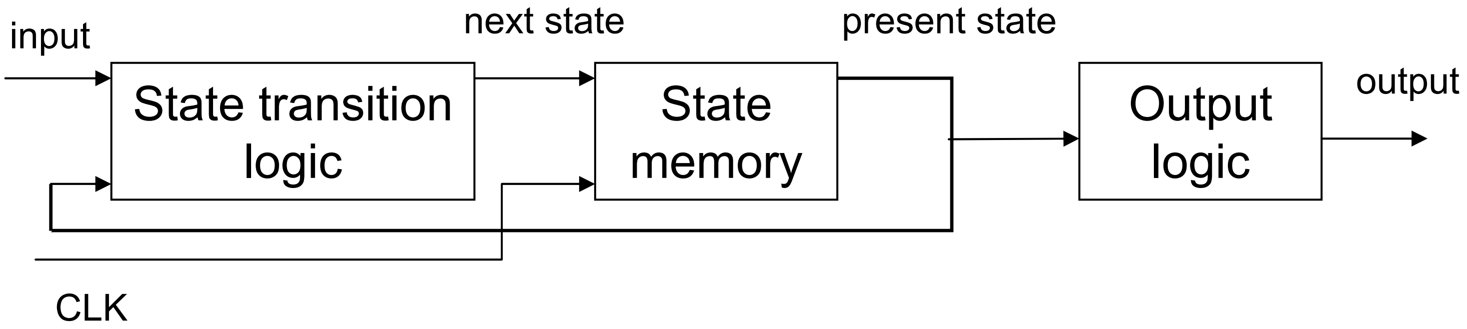
Decoder

1. case construction
   - Only for branching state of DFA
   - Problem with source code size and default case construction
   - Implemented with recursive backtracking
2. if construction
   - One for each path
   - Simple setting of invalid signal value
   - Implemented with preorder and with set of all paths

- Search for optimal traversal algorithm

Controller

- From DFA of events generate FSM representation in VHDL



- Connect controller signals with activation edges of Process graph

Conclusion

- The same internal model of controller and decoder for HDL generating as in simulator generating.
- Is possible some verification?
  - Architecture behavior in simulator
  - Architecture behavior in hardware