Lecture 2: High-Level Synthesis

CSCE 6730
Advanced VLSI Systems

Instructor: Saraju P. Mohanty, Ph. D.

NOTE: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.
Outline of Lecture

2. To discuss Design Automation terminology.
3. To know Data Flow Graph (DFG) / Control Flow Graph details (CFG).
4. To learn High-Level Synthesis (HLS) with the help of a small example.
5. To discuss Integer Linear Programming (ILP) based scheduling in details.
Note: Functional Block Level is RT Level
Synthesis Flow: Different Abstractions

1. System Specifications
2. System Level Design
   - Hardware/Software Allocation or Partitioning
3. Behavioral Description
4. High Level Synthesis
   - Transformation, Scheduling, Module Selection
5. RTL Description
6. Logic Synthesis
   - Two-Level, Multi-Level Synthesis
7. Gate Level Description
8. Layout Synthesis
   - Placement, Routing, Clock Distribution
9. Layout Level Description
Taxonomy .... ....

Concentric Circle: Levels of Abstraction

Axes: Domains of Description

Behavioral Domain: what the design is supposed to do ??

Structural Domain: one-to-one mapping of a behavioral representation to a set of components

Physical Domain: bind the structure to silicon
Taxonomy: Behavioral Vs Structural

Entity XOR2 is
Port( A, B: in bit; Z: out bit);
End XOR2;

-- This is Structural

Architecture DATAFLOW of XOR2 is
Begin
    Z <= A XOR B;
end DATAFLOW
Behavioral Vs Structural .......

-- This is Behavioral
Architecture ALGORITHM of XOR2 is
Begin
    Process(A, B)
    begin
        if A = B then
            Z <= ‘0’;
        else
            Z <= ‘1’;
        end if;
    end process
End ALGORITHM
Transition between domains

Behavioral Domain

Synthesis

Analysis

Optimization

Structural Domain

Refinement

Abstraction

Generation

Extraction

Physical/Geometrical Domain
What is High-Level Synthesis??

The high-level synthesis process is defined as a translation process from behavioural description to a structural description.

[Analogous to "compiler" that translates high-level language like C/Pascal to assembly language.]

Note: It is along the arc in Y-chart
Yet another definition of High-Level Synthesis……

McFarland (1990): (widely followed !!!)

“HLS is conversion or translation from an algorithmic level specification of the behavior of a digital system to a RT level structure that implements that behavior.”

Note 😞 It is along the **axis** in Y-chart
Why High-Level Synthesis?

- Shorter design cycle
- Fewer errors
- The ability to search the design space
- Documenting the design process
- Availability of IC technology to more people
High-Level Synthesis Phases

- HDL
- Compilation
- Transformation
- Scheduling
- Allocation / Binding
- Output Generation
- RTL Description
- Data Flow Graph
HLS Process .... ......

Compilation: Compile from VHDL to DFG/CFG (can be thought of as non-optimized compilation of programming language !!!)

Transformation: Initial CDFG is transformed so that the resultant CDFG is more suitable for following phases (similar to compiler optimization like dead code elimination !!!)
HLS Process: Compilation

\[ x = a + b; \]
\[ y = c - d; \]
\[ z = x \times y; \]
\[ y1 = b + d; \]

Compilation: Behavioral VHDL to structural VHDL and then draw the DFG

Note: CDFG more details coming soon……….
HLS Process: Transformation

\[ X^2 + AX + B \]
HLS Process: (Scheduling/Allocation)

**Scheduling:** partitions (with respect to: time) variables and operations in the DFG so that the operations in the same group can be executed concurrently.

**Allocation:** partitions with respect to hardware resources (functional units and memory units)

**Binding/Assignment:** assignment of operations to functional units and variables to memory units
Behavioral Synthesis: A small example

VHDL Code (Structural)

\[ Z := (X+Y) \times (E-F); \]

DFG

Step 1: Compilation and Transformation
HLS: Example ......

Step2: Scheduling (time/resource constraints)
HLS : Example ......

Step3: Allocation (fixes: amount and types of resources)
HLS: Example …

Step 4: Binding (which resource will be used by which operation)
HLS: Example ......

Step 5: Connection Allocation (communication between resources; bus, buffer/MUX)
HLS : Example …..

Datapath

CT1- Action: \( A = X + Y \)
Signals : Sel_A, Sel_B, load(Reg_A)

CT2- Action: \( B = E - F \)
Signals : Sel_A, Sel_B, load(Reg_B)

CT3- Action: \( Z = A \times B \)
Signals : load(Reg_z)

Control

Step6: Architecture Generation (Datapath and Control)
Control/Data Flow Graph (Formal Definition)

**DFG:** Graph $G = (V, E)$, where:

i. $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set whose elements are nodes, and

ii. $E \subset V \times V$ is an asymmetric data flow relation, whose elements are directed data edges.

**CFG:** Graph $G = (V, E)$, where:

i. $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set whose elements are nodes, and

ii. $E \subset V \times V$ is a control flow relation, whose elements are directed sequence edges.
Example (Sqrt Calculations):

\[ Y := 0.22 + 0.89 \times X; \]
\[ I := 0; \]
DO UNTIL I > 3 LOOP
    \[ Y := 0.5 \times (Y + X/Y); \]
    \[ I := I + 1; \]
END DO
DFG (sqrt example)
CFG (sqrt example)
Scheduling Algorithms

1. As-Soon-As-Possible (ASAP)
2. As-Late-As-Possible (ALAP)
3. Time Constrained Scheduling (such as: ILP, Force-directed heuristic method, Iterative refinement method)
4. Resource Constrained Scheduling (such as: Resource-based scheduling method, Static-list scheduling method)
5. Other Algorithms (like simulated annealing)
Scheduling Algorithms ……
(HAL Benchmark: Description)

\[ \frac{d^2 y}{dx^2} + 5 \frac{dy}{dx} x + 3y = 0 \]

while ( x < a ) do

\[ x_1 := x + dx; \]
\[ u_1 := u - (3 * x * u * dx); \]
\[ y_1 := y + (u * dx); \]
\[ x := x_1; \]
\[ u := u_1; \]
\[ y := y_1; \]

end while
Scheduling Algorithms ……
(HAL Benchmark: DFG)
Scheduling Algorithms ........ (ASAP Schedule)
Scheduling Algorithms ........ (ALAP Schedule)
Scheduling ........
(ILP Method: mobility)
Scheduling .......
(ILP Method: formulation)

\[ \text{minimize} \quad \text{total cost} = C_m * N_m + C_a * N_a + C_s * N_s + C_c * N_c \]

- \( C_m \) = cost of a multiplier, \( C_a \) = cost of an adder,
- \( C_s \) = cost of a subtracter, \( C_c \) = cost of a comparator
- \( N_m \) = number of multipliers, \( N_a \) = number of adders
- \( N_s \) = num. of subtracters, \( N_c \) = num of comparators

subject to:

(1) Operation \( o_i \) in state \( s_j \) \( (\sum_{E_i \leq j \leq L_i} x_{i,j} = 1) \)
- \( x_{1,1} = 1 \), \( x_{2,1} = 1 \), \( x_{3,1} + x_{3,2} = 1 \)  
- total 11 constraints for 11 operation (see mobility fig)
Scheduling ……. (ILP Method: formulation)…….

subject to: ………

(2) No control step contains more than \(N_{t_k}\) operations of type \(t_k\): \(\sum_{i \in \text{INDEX}_{t_k}} x_{i,j} \leq N_{t_k}\),
\(x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq N_m\) (multipliers in state 1) …..
total 12 constraints (for each state sum of the same type)

(3) For an operation \(o_j\) all predecessors should be scheduled in an earlier control step (if \(x_{i,k} = x_{j,l}\) then \(k < l\)): \(1x_{3,1} + 2x_{3,2} - 2x_{6,2} - 3x_{6,3} \leq -1\) ….
total 3 constraints.
Scheduling 
(ILP Method: solution)

Assuming:
\[ C_m = 2, \quad C_a = C_s = C_c = 1 \]

The cost function is minimized and all inequalities are satisfied when the values for the variables are:
\[ N_m = 2, \quad N_a = N_s = N_c = 1 \]
\[ x_{1,1} = x_{2,1} = x_{3,2} = x_{4,3} = x_{5,2} = x_{6,3} = x_{7,3} = x_{8,4} = x_{9,4} = x_{10,2} = x_{11,4} = 1 \quad \text{and rest all} \quad x_{i,j} = 0. \]
Scheduling .......
(ILP Method: schedule)