VLSI design

P.J. Drongowski
SandSoftwareSound.net

Copyright © 1990-2013 Paul J. Drongowski
Automated synthesis


Synthesis

- Mapping a behavioral specification to a hardware implementation while meeting a set of goals and constraints
- Advantages
  - Produce and update a design quickly
  - Generate a range of implementations
  - Correct by construction (synthesizer must be verified!)
  - Produce implementations to check manual designs
  - Ability to "search the design space"
  - Track design decisions and justifications
  - Make IC technology available to more people
- Three categories of synthesis programs
  - Algorithm
    ◦ Input: Abstract behavior
    ◦ Output: Behavior with control flow
  - Register transfer
    ◦ Input: Behavior with control flow
    ◦ Output: Register transfer structure
  - Logic
    ◦ Input: RT structure and detailed control flow
    ◦ Output: Gate level structure or layout
    ◦ Two major approaches
      ◦ Decompose into primitive elements (gates)
      ◦ Match registers / operators against library
- Typical process flow (sequence)
  ◦ Algorithm ➔ register transfer ➔ logic
Synthesis

- Input specification
  - Behavioral information
  - What functions to perform

- Target implementation
  - Behavioral specification
  - Structural information
  - Detailed behavior of the structure
  - Relationship between these three elements

- Goals
  - Maximum speed
  - Minimum cost (e.g., area, package count)
  - Minimum pin count
  - Minimum power consumption
  - Minimum design time
  - Maximum reliability
  - Maximum testability

- Constraints
  - Time delays for and between events
  - Area or package count upper bounds
  - Maximum number of pind
  - Upper bound on power consumption
  - Upper bound on software runtimes
  - Lower bound on reliability
  - Lower bound on testability
Subproblems

- Resource allocation
  - Select structures to implement functions
  - Many-to-many mapping
- Design transformation
  - Change design to achieve goal or meet constraint
  - Example
    - Delete a (temporary) buffer register
    - Share use of a pre-existing register instead
- Decomposition (composition)
  - No direct mapping from function to structure
  - Decompose function until mapping is direct
  - Composite structure built on realizable primitives
- Event scheduling
  - Assign each operation to a time slot
  - Slot may be a clock phase or interval

Approaches

- Synthesize and improve
  - Generate a correct solution
  - Transform solution to optimize objectives
- Optimal synthesis
  - Perform optimization during synthesis
  - If criteria are not met, synthesize a new design
- Programming techniques
  - Heuristic
  - Mathematical programming formulations
  - Expert systems
Algorithm synthesis

- Input specification, goals and constraints
- Output is specification with complete control flow info
- Input specification
  - "Black box" behavior
  - A list of input variables
  - A list of output variables
  - Functions performed on inputs to produce outputs
  - Partial orderings on functions
    - Data precedence
    - External constraints between
      - Reading inputs and
      - Writing outputs
- Description of variables
  - Specified bit width
  - Binary notation
  - May be subdivided into fields
  - Increases complexity of bookkeeping
- Internal representations
  - Parse trees
  - Graphs (most popular)
    - Control flow (order of events)
    - Data flow (producer - consumer)
- High level transformations
  - Detection (removal) of common subexpressions
  - Constant folding and propagation
  - Dead code elimination
  - Inline expansion of procedures
  - Loop unrolling
  - Serial - parallel design decisions
- Local transformations
  - Change loop ending criterion
  - Division by 2 to right shift
  - Multiplication by 2 to left shift
  - Increment / decrement
Algorithm (example)

- Compute square root of X using Newton's method
- Number of iterations needed is usually small (e.g., 4)
- Initial value given by 1st degree minimax polynomial

\[
Y := 0.222222 + 0.888889 \times X ;
I := 0 ;
\text{do until } I > 3 \text{ loop}
\quad Y := 0.5 \times (Y + X / Y ) ;
\quad I := I + 1 ;
\text{endo} ;
\]

- No dependence between "I+1" and calculation of Y
- Dataflow arc represents producer - consumer relationship
- Possible local transformations
  - Loop-ending criterion changed to "I = 0"
  - Use two bit variable for I
  - Replace "* 0.5" by right shift
  - Replace addition of 1 to I by increment
- Could unroll loop due to small number of iterations
Optimized example

• Graphs (below) depict effect of local transformations
  • Trivial solution
    • Uses one functional unit and one memory
    • Requires 23 control steps
  • Parallel solution (shown below)
    • Constrained only by essential dependencies
    • Two dummy nodes delimit the loop boundaries
    • Two functional units are required
    • Only $2 + (4 \times 2) = 10$ control steps needed
Register transfer synthesis

- Two major activities
  - Generate datapath structures
  - Synthesize control (sequence events in datapath)
- Both activities should proceed in parallel
  - Resource sharing ⇒ more complex control sequences

Datapath synthesis

- Assume control flow is (at least) partially specified
- Behavior is given in dataflow or register transfer form
- Allocation
  - Goal: minimize amount of hardware needed
  - Fn'l units, memory, paths usually minimized separately
  - Variables (values) are allocated to registers
    - Assign value to register
    - If it is generated in one control step
    - And used in a later (different) control step
    - Assign to same register if lifetimes do not overlap
  - Operations are allocated to operators
    - Share fn'l unit if operations are in different time steps
    - Problem
      - Group mutually exclusive operations such that
      - Minimum number of groups is obtained
  - Registers and operators are interconnected
  - Multiplexers are inserted as required
- Final event scheduling
  - Minimization goals
    - Length (duration) of time step
    - Number of time steps to perform function
  - Determine resource sharing
  - Schedule register transfers into time slots
  - Generate timing analysis
    - Compute propagation delay
    - Determine / report set-up and hold times
Datapath allocation

• Iterative / constructive techniques
  • Assign elements one at a time
  • Generally look at less of the search space (than global)
    ◦ Usually more efficient
    ◦ But, less likely to find optimal solution
• "Global" selection of element for assignment
  ◦ Select item using some metric
  ◦ Example: Item that minimizes increase in cost
  ◦ Minimize fn'l units, registers, MUX's [EMUCS]
  ◦ Minimize interconnect [Elf]
  ◦ Earliest value in dataflow [REAL]
• "Local" selection of element
  ◦ Select items in a fixed order (e.g., greed)
  ◦ In order of occurrence in dataflow graph [Hafer]
  ◦ Expert knowledge [Kowalski's DAA]

• Global allocation techniques
  • Simultaneous solution to multiple assignments
  • Graph theoretic formulation
    ◦ Objects to be assigned are nodes
    ◦ Arc between nodes indicates possible sharing
    ◦ Find sets of nodes where
      ◦ Members are connected to one another
      ◦ Members in set can share without conflict
    ◦ Clique finding problem
      ◦ Find minimum # of cliques that cover graph
      ◦ Find maximum cliques in the graph
      ◦ Problem is NP-hard (try heuristics)
  • Mathematical programming
    ◦ Create variable for each possible assignment
    ◦ Variable = 1 if the assignment is made, else 0
    ◦ Find solution that minimizes cost function
    ◦ Constraints to one - to - one mapping of objects
    ◦ Optimal solution requires exhaustive search!
Scheduling
  • Interaction between scheduling and allocation activities
  • Type of scheduling algorithm

Interaction with allocation
  • Vicious circle between allocation and scheduling
    • Can two operations be scheduled into same step?
      ◊ Do they use the same functional unit?
    • What are op delays so efficient schedule can be found?
      ◊ What is delay of fn'l units and interconnections?
    • How many fn'l units and what allocation of operators?
      ◊ Which operations should be done in parallel?
  • Set (no) limit on the number of fn'l units, then schedule
    • FACET, DAA, Flamol: Let user specify limit
    • MIMOLA: Iteratively adjust limit and reschedule
    • Chippe: Use expert system and allocator feedback
  • MAHA [Parker]
    • Develop schedule and resource needs simultaneously
    • Allocate functions as it schedules
    • Adds fn'l units when it cannot share existing ones
  • HAL [Paulin]
    • Force directed scheduling
    • Schedules without time constraint
    • Balances number of functional units per step
    • Number req'd is maximum number req'd in any step
    • Can reschedule after detailed datapath design
  • Yorktown Silicon Compiler
    • Assign each operation to its own unit
    • Perform all operations in one control step
    • Iteratively increase sharing
  • BUD
    • Cluster operations using metric
    • Metric accounts for sharing, interconnect, parallelism
    • Assign functional unit to each cluster
Scheduling algorithms

- Two basic classes
- Transformational
  - EXPL
    - Exhaustive search
    - Try all S-P transformations and find best one
    - Could be improved by branch and bound
    - Cut off suboptimal search path
  - CAMAD and YSC
    - Heuristic search
    - Move design closer to goal
    - YSC produces pastest possible schedule
- Iterative / constructive
  - Add operations until all are scheduled
  - Differences
    - How to choose the next operation
    - Where to schedule each operation
  - ASAP (CMUDA, MIMOLA, Flamel)
    - Assumes fn'l units have been specified
    - Sort operations topologically
    - Schedule each operation in order
    - No priority is given to operations on critical path
    - Less important operations can block critical ones
  - List scheduling (Elef, BUD, ISYN)
    - Sort list by priority
    - Schedule highest priority op until resources exhausted
    - Move to next control step and repeat
  - Freedom-based scheduling
    - Schedule critical path operations first
    - Next, schedule operations with least freedom first
    - Smallest range of possible control steps
  - Force-directed scheduling [Paulin]
    - Range of possible steps forms distribution graph
    - Graph shows how heavily loaded each step is
    - Op is selected and placed to balance distribution
Control synthesis

- Selection of clocking scheme
  - Determine number and length of clock phases
  - Allocate events to clock phases
  - Produce revised, synchronized control flow
- Controller design style is often preselected
  - Random logic
  - Microcode
  - Programmable logic array (PLA)

Logic synthesis

- Decompose into primitive elements (gates and flip/flops)
- Many input formats
  - Logic equations
  - Truth table
  - State table (for finite state machines)
- Perform gate-level optimizations (Boolean algebra)
- Control synthesis
  - Often PLA based
  - Must generate microcode
  - Techniques (and trades)
    ◊ Vertical vs. horizontal microcode
    ◊ Encoding to reduce space (code size)
    ◊ "Fold" PLA to reduce size

Module binding

- Alternative to logic synthesis
- Match operator / register requirements against library
- Find element that most nearly meets requirements
- Advantages
  - Target components are known and characterized
  - Size and timing estimation is more accurate
- May disallow solution using custom hardware
Subproblems

• Resource allocation
  • Select structures to implement functions
  • Many-to-many mapping
• Design transformation
  • Change design to achieve goal or meet constraint
  • Example
    ◇ Delete a (temporary) buffer register
    ◇ Share use of a pre-existing register instead
• Decomposition (composition)
  • No direct mapping from function to structure
  • Decompose function until mapping is direct
  • Composite structure built on realizable primitives
• Event scheduling
  • Assign each operation to a time slot
  • Slot may be a clock phase or interval

Approaches

• Synthesize and improve
  • Generate a correct solution
  • Transform solution to optimize objectives
• Optimal synthesis
  • Perform optimization during synthesis
  • If criteria are not met, synthesize a new design
• Programming techniques
  • Heuristic
  • Mathematical programming formulations
  • Expert systems