

# Optimization for Embedded Systems Platforms

CSE594  
Jennifer Wong  
10/14/08

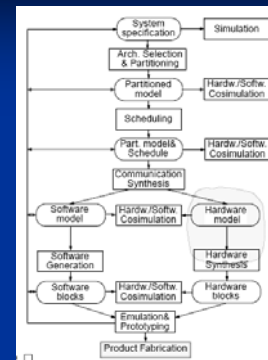
## Embedded Systems

- **Complexity:** new computer aided methodologies are needed in the context of increasing complexity (SoC).
- **Verification:** starting from a (formal) system specification which is validated, and performing well defined design steps (transformations) with verifiable outputs.
- **Time to market:** only efficient tools and reuse can bring design productivity up to the expected level.

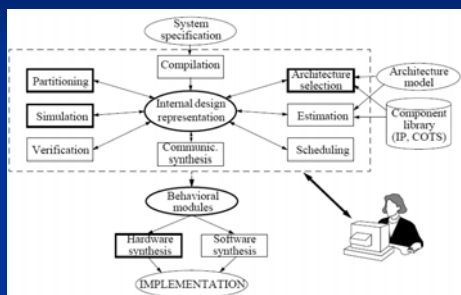
## System Synthesis

- **Input:** an implementation independent specification of the system; this includes: functionality and constraints.
- **The synthesis tasks:**
  - To select the architecture
  - To partition functionality over the components of the architecture
  - To schedule activities
  - To generate behavioral modules corresponding to the hardware and software domain of the implementation, including interface modules
  - The behavioral modules resulted from the previous steps are further synthesized into the actual hardware and/or software implementation

## System Synthesis



## System Synthesis



## Synthesis Steps

- **Synthesis:**
  - Transformation of a representation in the **behavioral domain** to a representation of the same design in the **structural domain** (at the same abstraction level).
  - The structural description which results after a synthesis step is formulated as an **interconnection of abstract components**.
  - Each such component is functionally specified at the following, lower abstraction level. These functional specifications are the input for the following synthesis step.

## Synthesis Steps

- **System synthesis**
  - Input: System level specification (interacting processes) + design constraints
  - Output: Behavioral elements to be synthesized to hardware and software + System architecture + Process schedule and mapping
- **High level (behavioral) synthesis**
  - Input: Algorithmic description
  - Output: RT level description of the controller (FSM) + net-list (data path)

## Synthesis Steps

- **RT level synthesis**
  - Input: RT level description of the controller (FSM) + net-list
  - Output: Blocks of combinational and memory elements
- **Logic synthesis**
  - Input: Blocks of combinational and memory elements (as boolean functions)
  - Output: gate-level net-list
- **Physical design**
  - Input: gate-level netlist
  - Output: geometrical layout for a given technology

## Introduction to Integer Linear Programming

## Linear Programming

$$\max \sum_{i=1}^n c_i x_i$$

$$\text{Subject to } Ax \leq b$$

## ILP Formulations

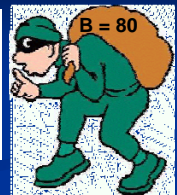
- Mixed Integer Linear Programs (MIP)
- Integer Linear Programs (ILP)
- 0-1 Integer Linear Programs (0-1 ILP)
  
- Objective Functions
- Constraints
- Variable Declarations

## Knapsack Problem

W = 1, U = 100



W = 75, U = 250



W = 3, U = 100



W = 20, U = 1000



W = 1, U = 500

W = 15, U = 600

## Knapsack Problem

ITEM	Weight (W <sub>i</sub> )	Utility (U <sub>i</sub> )
1. Watch	1	100
2. Necklace	1	500
3. Candlesticks	3	100
4. TV	75	250
5. Laptop	15	600
6. Cash	20	1000

## Knapsack Problem

- Variables

$$x_i = \begin{cases} 1, & \text{if item is selected} \\ 0, & \text{if item not selected} \end{cases}$$

- Objective Function

$$\max \sum_{i=1}^n u_i x_i$$

## Knapsack Problem

- Constraints

$$\sum_{i=1}^n w_i x_i \leq B$$

- Variable Types

$$0 \leq x_i \leq 1$$

## Knapsack Problem

- Objective Function

$$\max(100x_{\text{watch}} + 500x_{\text{necklace}} + 100x_{\text{candlestick}} + 250x_{\text{TV}} + 600x_{\text{laptop}} + 1000x_{\text{cash}})$$

- Constraints

$$(1x_{\text{watch}} + 1x_{\text{necklace}} + 3x_{\text{candlestick}} + 75x_{\text{TV}} + 15x_{\text{laptop}} + 20x_{\text{cash}}) \leq 80$$

- Variables

$$\begin{aligned} 0 \leq x_{\text{watch}} &\leq 1 & 0 \leq x_{\text{TV}} &\leq 1 \\ 0 \leq x_{\text{necklace}} &\leq 1 & 0 \leq x_{\text{Laptop}} &\leq 1 \\ 0 \leq x_{\text{candlesticks}} &\leq 1 & 0 \leq x_{\text{cash}} &\leq 1 \end{aligned}$$

## Solvers

- Public Software: LP\_SOLVE

- Introduction

<http://www.geocities.com/lpsolve/>  
<http://lpsolve.sourceforge.net/5.5/>

- Download

<http://www.cs.sunysh.edu/~algorithm/Implement/lpsolve/Implement.shtml>

- Web Applet

<http://not-reor.berkeley.edu/not/Applications/SimplexDemo/Simplex.html>

- Commercial Software: CPLEX

<http://trix.cs.ucla.edu/jenni/CPLEX/>

## File Formats

- LP format

- Objective function required and first
- Semi-colon required on each line (;)
- Comments allowed /\* \*/
- Variables must start with a letter, can contain \_[]{}./&#;%~@^
- Relational Operators: <, <=, =, >, >=
- All variables on left side of equation, constants only on the right side

## Knapsack Problem: LP Format

```
max : 100xW + 500xN + 100xC + 250xT + 600xL + 1000x$;  
xW + xN + 3xC + 75xT + 15xL + 20x$ ≤ 80;  
xW ≤ 1;  
xN ≤ 1;  
xC ≤ 1;  
xT ≤ 1;  
xL ≤ 1;  
x$ ≤ 1;  
int xW, xN, xC, xT, xL, x$;
```

## Lp\_Solve

```
# lp_solve < knapsack.txt
```

Options:

- d provides debug information
- Sprints solution
- h help

## Lp\_solve Solution Format

Value of objective function:	2300
x\$	1
xC	1
xL	1
xN	1
xT	0
xW	1

## File Formats

- CPLEX
  - Objective function required (MINIMIZE, MAXIMIZE, MINIMUM, MAXIMUM, MIN or MAX)
  - Comments (\)
  - Variables < 16 characters
  - Must state SUBJECT TO before constraints
  - Each constraint on a new line
  - Must label BOUNDS section
  - END label at end of the problem
- LP Format for CPLEX
  - [http://plato.asu.edu/cplex\\_lp.pdf](http://plato.asu.edu/cplex_lp.pdf)

## Knapsack Problem: CPLEX Format

```
MAXIMIZE  
100xW + 500xN + 100xC + 250xT + 600xL + 1000x$  
SUBJECT TO  
xW + xN + 3xC + 75xT + 15xL + 20x$ ≤ 80  
BINARY  
xW  
xN  
xC  
xT  
xL  
x$  
END
```

## CPLEX

```
# read knapsack.lp  
# optimize  
# write knapsack.mst  
# quit
```

## CPLEX Solution Format

Tried aggregator 1 time.  
MIP Presolve modified 2 coefficients.  
Reduced MIP has 1 rows, 6 columns, and 6 nonzeros.  
Presolve time = 0.00 sec.  
Clique table members: 2  
MIP emphasis: optimality  
Root relaxation solution time = 0.00 sec.

Integer optimal solution: **Objective = 2.3000000000e+03**  
Solution time = 0.00 sec. Iterations = 1 Nodes = 0

```

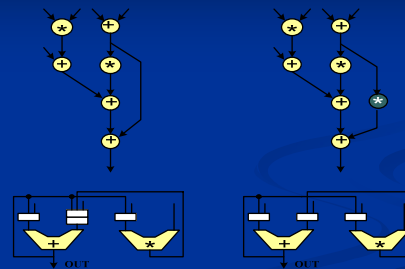
NAME:      knapsack.cplex   MIP Start
xW         1
xN         1
xC         1
xF         0
xL         1
xS         1
ENDATA
    
```

## Minimizing Global Interconnect using Bypassing and Chaining

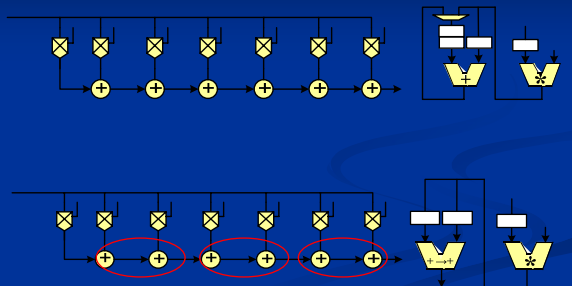
### Focus of Behavioral Synthesis

- Traditional Behavioral Synthesis
  - Emphasis on minimizing area of execution units and registers
  - Scheduling, assignment, graph coloring-based register assignment
- Behavioral Synthesis for Deep Submicron
  - Emphasis on minimizing interconnect
  - Interaction with architecture (bypassing and chaining)

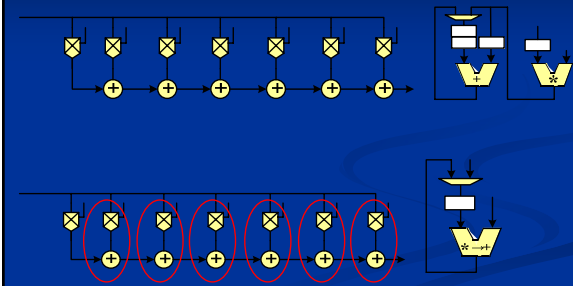
### Bypassing Operations - Motivation



### Chaining Operations - Motivation



### Chaining Operations - Motivation



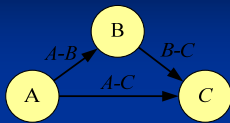
## Objective

- To develop synthesis techniques that consider the means of architecture exploration in order to address the needs of deep submicron interconnect.
- Architecture -> bypassing and chaining
- Needs of deep submicron -> reduction of interconnect
- Even in traditional synthesis, MUXs begin to dominate overall area

## Outline

- Bypassing
  - Concept, problem formulation, complexity, ILP-based solution, heuristic approach
- Chaining
  - Concept, problem formulation, complexity, ILP-based solution, heuristic approach
- Experimental Results

## Bypass Operations

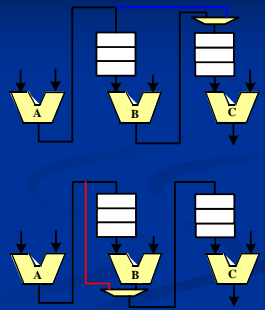


### Advantages:

- Reduction interconnect
- Reduction in MUX
- No increase in critical path
- Possible decrease in clock cycle time

### Disadvantages:

- Possible increase in clock cycle time
- Increase area (registers for constants)



## Addition of Bypass Operation

**Problem:** Maximal Incoming Interconnect per Functional Unit

### Instance:

- architecture constraints over a set of functional units
- maximum incoming interconnect for each functional unit
- CDFG
- usage frequency for each transfer operation
- limitation on the number of bypass operations

### Question:

- Schedule for CDFG s.t.
  - maximum number of incoming interconnect per unit is ensured
  - using less than allowed bypass operations

## ILP

- Constants
  - # of incoming interconnects for ea unit
  - # of transfers of each type
- Variables
  - Which interconnects to remain
  - Type of bypass operation to added to a transfer type
  - If a transfer is used in conjunction with a bypass operation

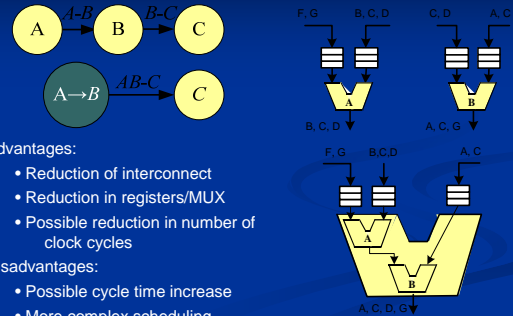
## ILP

- Constraints
  - A bypass must be added to every transfer (can be unit identical to transfer ends)
  - A transfer must remain if
    - used in conjunction with a bypass operation
    - no bypass was applied to the transfer
  - No more than specified number of interconnect remain per unit
- Objective function
  - Minimize # of added bypass operations

## Heuristics

- More comprehensive formulation
  - Intermediate benefit and future prospects
    - Weighted sum of components
  - Accurate picture – updating
    - Frequency of interconnect use
  - Statistical tuning of parameters
    - Modify weight factors as become closer to solutions
- Scheduling difficulty

## Chaining Operations



### Advantages:

- Reduction of interconnect
- Reduction in registers/MUX
- Possible reduction in number of clock cycles

### Disadvantages:

- Possible cycle time increase
- More complex scheduling

## Chaining Operations

- **Problem:** Chaining for Long Interconnect Reduction
- **Instance:**
  - architecture constraints over a set of functional units
  - maximum total number of incoming interconnect
  - maximum incoming interconnect for each functional unit
  - maximum allowable incoming interconnect for each chain unit
  - CDFG
  - usage frequency for each interconnect
  - limitation on the number of chaining operations
- **Question:**
  - Schedule for CDFG st.
    - maximize number of chaining operations are used

## ILP

- Constants
  - # of incoming interconnect for ea unit
  - # of incoming interconnect for ea chained unit
  - # of total interconnect
  - # of transfers of each type
  - Which transfers can be chained

## ILP

- Variables
  - Which units are selected to chain
  - If two consecutive operations are chained
- Constraints
  - Chains are created in one direction  $i \rightarrow j$
  - Only allowable FU can be chained
  - # incoming to each unit, chain, and total must be less than allowable
- Objective function
  - Maximize # of chained operations

## Experimental Results

Benchmark	Most Common Transfers
Square Root	add <sub>u</sub> →add (768), add→ld <sub>f2</sub> (732), add→pred <sub>lt</sub> (798)
Viterbi Encoder	ld <sub>i</sub> →xor (1400), add <sub>u</sub> →add (1500), add→ld <sub>i</sub> (1300)
g721 decode	mov→ld <sub>i</sub> (189758), mov→st <sub>i</sub> (103504), lsl→or (17250)
g721 encode	add→pred <sub>lt</sub> (14914), ld <sub>i</sub> →add (22373), mov→ld <sub>i</sub> (82022)

## Experimental Results - Bypassing

Benchmark	Orig # IC	At most 3 In. IC	% IC Reduction	Runtime Overhead
Square Root	56	17	69%	0.6%
Convolution Encoder	42	10	76%	1.3%
g721 encode	72	25	65%	3.1%
g721 decode	72	21	70%	2.1%

Benchmark	Orig # IC	At most 3 In. IC	% IC Reduction	Runtime Overhead
Square Root	56	18	67%	0.1%
Convolution Encoder	42	12	71%	0.2%
g721 encode	72	26	63%	0.7%
g721 decode	72	32	55%	0.7%

## Experimental Results - Chaining

Benchmark	Orig # IC	Cycle Time	% IC Reduction	Runtime Reduction
Square Root	56	12	78%	1.0%
Convolution Encoder	42	9	78%	3.4%
g721 encode	72	19	73%	22.6%
g721 decode	72	17	76%	14.3%

Benchmark	Orig # IC	Cycle Time	% IC Reduction	Runtime Reduction
Square Root	56	15	73%	2.7%
Convolution Encoder	42	9	78%	3.4%
g721 encode	72	21	70%	35.8%
g721 decode	72	19	73%	20.7%

## Conclusion

- Behavioral synthesis driven by architecture concepts and needs of deep submicron
- Minimizing MUXs and interconnect is prime target
- Optimal and heuristic treatment of bypassing and chaining

## Synthesis of ES

- Set cover
- Scheduling
- Boolean Satisfiability
- Register Assignment
  - Graph coloring