Balancing FPGA Resource Utilities

Xuejun Liang
Department of Computer Science, Jackson State University
Jeffrey S. Vetter, Melissa C. Smith, and Arthur S. Bland
Oak Ridge National Laboratory, Oak Ridge

Heterogeneous Components in Xilinx® Virtex-II FPGA

- CLB/Slice
  - Lookup Table
    - Logic Function
    - Slice Distributed RAM
    - Slice SRL
  - Register
- Block SelectRAM
- Embedded Multiplier

TASK
is
Balancing
Utilizations
of

- CLB/Slice
- Block SelectRAM
- Embedded Multiplier

Main Topic and Methodology

- Main Topic: Balancing Utilizations of
  - CLB/Slice
  - Embedded Multiplier
- Methodology
  - Creating Multipliers with Different Combinations of
    - CLB/Slice, and
    - Embedded Multiplier
  - Selecting a Proper Multiplier for a Given Multiplication Operation in an Application such that the Utilizations of CLB/Slice and Embedded Multiplier are balanced.

Outlines

- Related Work
- SRC-6E Platform Overview
- Motivating Example
- Problem Formulation
- Two Algorithms for Solving the Problem
- Experiment Results

Related Work

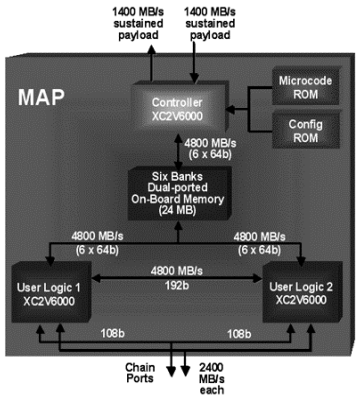
- A Novel 2D Filter Design Methodology For Heterogeneous Devices, FCCM 2005
  - Using Singular Valve Decomposition to approximate a 2D Filter with a number of 2X1D convolutions and one low complexity 2D convolution. This reduces the number of high-precision multipliers required for implementation.
- Migrating Functionality From ROMs to Embedded Multipliers, FCCM 2004
  - Using uniform piecewise polynomial approximation.

Related Work (Cont.)

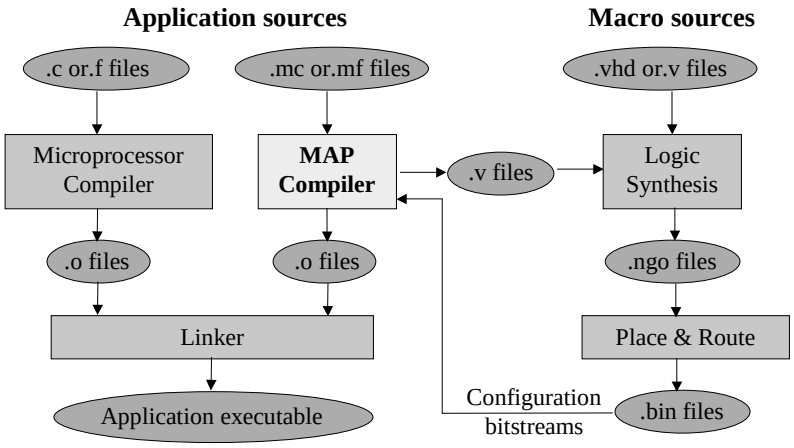
- Register Binding for FPGAs with Embedded Memory, FCCM 2004
  - Mapping variables to embedded RAMs rather than registers. This results in large saving in FPGA slices at the expense of increased use of FPGA embedded RAMs.
- SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays,
  - Identifying part of the circuit that can be implemented in embedded RAMs

SRC-6E Platform Overview Hardware Architecture

- Dual-Microprocessor Board
- MAP® Processor
  - Two Xilinx® Virtex-II FPGA Chips
  - Six 4MB Banks of On-Board Memories
- Connected via SNAP® Card Which Plug Into the DIMM Slot on the Microprocessor Board.



SRC-6E Platform Overview Programming Model



Motivating Example

- FFT Computation Over an Array of Complex Vectors
- FFT Function Unit
  - Computing FFT over one complex vector
  - Using one on-board bank of memory
- Three FFT Function Units are Expected to Put on an FPGA Chip.
- But, They Do not Fit!

- FPGA Resource Utilities of the FFT Function Unit

FPGA Resources	Utilities	
Slices (33,792)	8,846	26%
Block RAMs (144)	6	4%
MULT18×18s (144)	52	36%

Motivating Example (Cont.)

- Balancing the FPGA Resource Utilities
  - A slice-based multiplier macro is implemented.
  - Four multiplications are allocated to the slice-based multiplier macros.
  - Three FFT function units fit on one FPGA chip.
- FPGA Resource Utilities

FPGA Resources	One Unit		Three Units	
Slices (33,792)	9,816	29%	26,098	77%
Block RAMs (144)	6	4%	18	12%
MULT18×18s (144)	40	27%	120	83%

Problem Formulation

Enumerating Related Multiplier Macros

- Type A: 32bit×32bit → 32bit
  - Multiplier Macros MA0, MA1, MA2, and MA3 with 0, 1, 2, and 3 MULT18×18s, respectively
- Type B: 32bit×16bit → 32bit
  - Multiplier Macros MB0, MB1, and MB2 with 0, 1, and 2 MULT18×18s, respectively
- Type C: 16bit×16bit → 32bit
  - Multiplier Macros MC0 and MC1 with 0 and 1 MULT18×18, respectively

Problem Formulation

Notations

Notation	Meaning
SA_n(n=0..3)	FPGA slices used by Macro MAn (n=0..3)
SB_n(n=0..2)	FPGA slices used by Macro MBn (n=0..2)
SC_n(n=0..1)	FPGA slices used by Macro MCn (n=0..1)
NA_n(n=0..3)	Number of Macro MAn (n=0..3) used
NB_n(n=0..2)	Number of Macro MBn (n=0..2) used
NC_n(n=0..1)	Number of Macro MCn (n=0..1) used
NA	Number of Type A multiplications
NB	Number of Type B multiplications
NC	Number of Type C multiplications

Problem Formulation

Notations (Cont.)

Notation	Meaning
TNS	Total number of FPGA slices
TNBM	Total number of block multipliers
TNBR	Total number of block RAMs
NBM	Number of block RAMs used
NBM18	Number of block RAMs used with up to 18-bits
NBM36	Number of block RAMs used with 36-bits
NS4I	Number of FPGA slice used for the interface
NS4C	Number of FPGA slice used for the computation

Problem Formulation

Conditions and Allocation Constraints

Conditions

$$SA_0 > SA_1 > SA_2 > SA_3 \geq 0$$
$$SB_0 > SB_1 > SB_2 \geq 0$$
$$SC_0 > SC_1 \geq 0$$

Allocation Constraints

$$NA_n \geq 0, n = 0, 1, 2, 3$$
$$NB_n \geq 0, n = 0, 1, 2$$
$$NC_n \geq 0, n = 0, 1$$
$$NA_0 + NA_1 + NA_2 + NA_3 = NA$$
$$NB_0 + NB_1 + NB_2 = NB$$
$$NC_0 + NC_1 = NC$$

Problem Formulation

Balance Constraints

$$CComp(NS4C) = Floor\left(\frac{TNS - NS4I}{NS4C}\right)$$

$$CMult(NBM) = Floor\left(\frac{TNBM}{NBM + NBR36}\right)$$

Balance Constraint $CMult(NBM) = CComp(NS4C)$

Slice-bound Constraint $CMult(NBM) \geq CComp(NS4C)$

Block multiplier-bound Constraint $CMult(NBM) \leq CComp(NS4C)$

Problem Formulation

P1 Determine NA_n (n=0, 1, 2, 3), NB_n (n=0, 1, 2), and NC_n (n=0, 1) to minimize *NS4C* and subject to the allocation constraint and the balance constraint

P2 Determine NA_n (n=0, 1, 2, 3), NB_n (n=0, 1, 2), and NC_n (n=0, 1) to minimize *NS4C* and subject to the allocation constraint and the slice-bound constraint

Problem Formulation

Q1 Determine NA_n (n=0, 1, 2, 3), NB_n (n=0, 1, 2), and NC_n (n=0, 1) to minimize *NBM* and subject to the allocation constraint and the balance constraint

Q2 Determine NA_n (n=0, 1, 2, 3), NB_n (n=0, 1, 2), and NC_n (n=0, 1) to minimize *NBM* and subject to the allocation constraint and the block multiplier-bound constraint

Problem Formulation

Propositions

- Minimizing *NS4C* implies maximizing *CComp(NS4C)*
- Minimizing *NBM* implies maximizing *CMult(NBM*)
- If *(P1)* has a solution to a problem, then it must be the solution of *(P2*) to the same problem.
- If (*Q1*) has a solution to a problem, then it must be the solution of *(Q2*) to the same problem.
- If both (*P2*) and (*Q2*) have solutions to the same problem, then both solutions must satisfy the balance constraint, and then the solution of (*P2*) is also the solution of (*P1*) and the solution of (*Q2*) is also the solution of (*Q1*).

Two Algorithms

- Naïve algorithm: Exhaustive search
  - Complexity: $(NA+1)^3 \times (NB+1)^2 \times (NC+1)^1$
- Greedy algorithm for solving *(P2)*
  - Initial condition

$$NA_0 = NA_1 = NA_2 = 0 \text{ and } NA_3 = NA$$
$$NB_0 = NB_1 = 0 \text{ and } NB_2 = NB$$
$$NC_0 = 0 \text{ and } NC_1 = NC$$

  - Basic idea: Decrease NBM by one at one step and keep NS4C with a minimal increase at each step until *CMult*(*NBM*) ≥ *CComp*(*NS4C*).

Experiment Results

Constant and initial values

TNS	**33792**	***NA***	**15**	SB_0	**257**
TNBM	**144**	***NB***	**3**	SB_1	**124**
TNBR	**144**	***NC***	**1**	SB_2	**62**
$NS4C_0$	**8141**	SA_0	**355**	SC_0	**163**
NS4I	**1675**	SA_1	**236**	SC_1	**57**
NBR18	**0**	SA_2	**185**		
NBR36	**6**	SA_3	**118**		

Experiment Results

Parameter	Naïve		Greedy P2	Manual
	P1 or P2	Q1 or Q2		
NA_0	0	0	0	4
NA_1	5	1	3	0
NA_2	0	14	1	0
NA_3	10	0	11	11
NB_0	0	2	0	0
NB_1	0	1	3	0
NB_2	3	0	0	3
NC_0	0	0	0	0
NC_1	1	1	1	1
NS4C	8731	9649	8748	9089
NBM	42	31	42	40
Max Copies	3	3	3	3
Complexity	795906	795906	318	N/A

Conclusions

- An FPGA module selection problem is formulated to deal with balancing FPGA resources.
- An efficient greedy algorithm to solve the problem are provided.
- More work should be done in Future