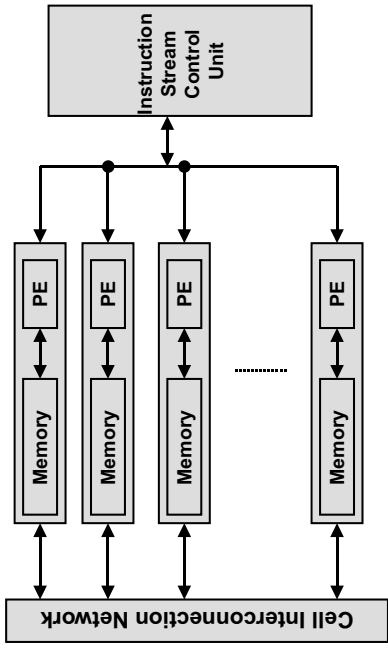
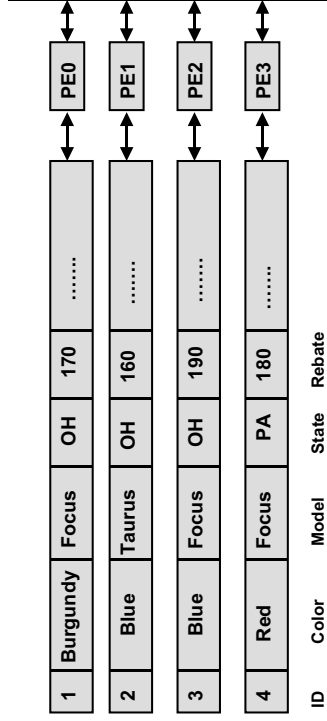


Implementing An Associative Processor on FPGAs

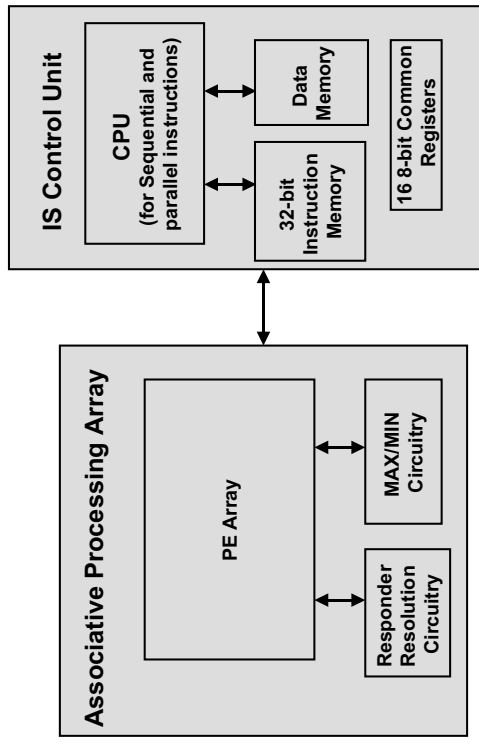
A Conceptual View of the KSU ASC Model



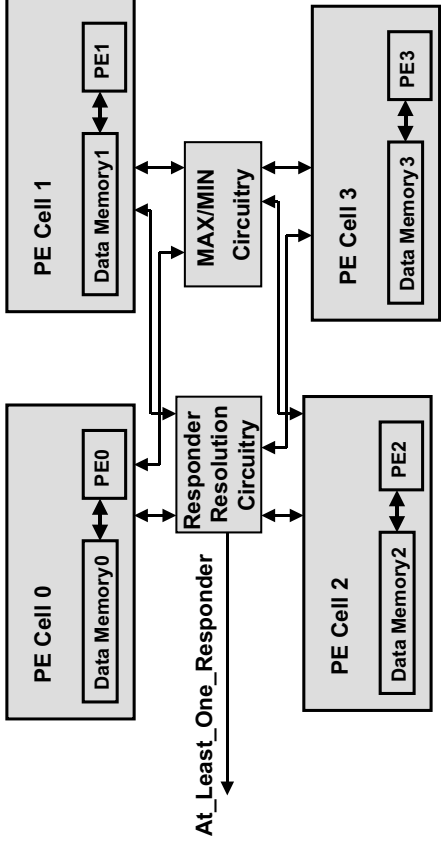
An Example for the Data Memory Organization: Auto Information Stored in the PE Cells



The Prototype of the Byte-serial ASC Processor

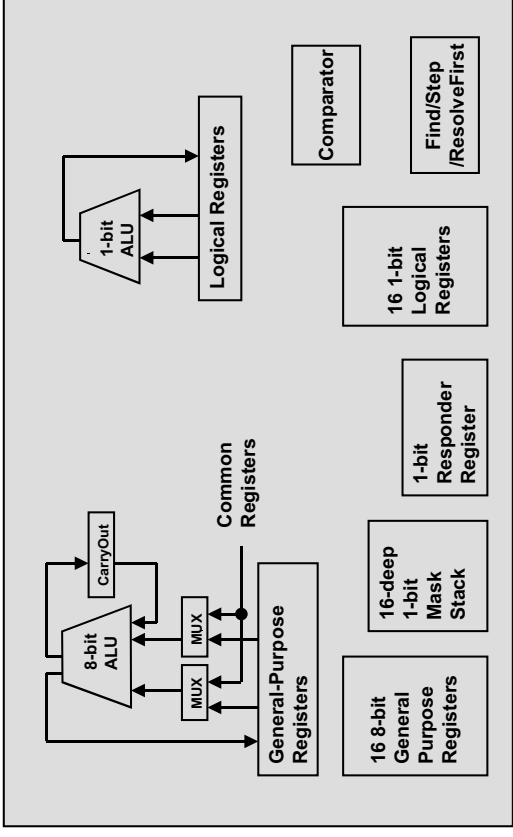


Prototype of the 4-PE Associative Processing Array



5

A Processing Element Overview



6

Instruction Set and Assembling Language (1)

- Data Transfer Instructions
 - LD *address, dstreg* - LDI *immediate, dstreg*
 - LDRR *srcreg, dstreg* - LDRRSPD *srcreg*
 - ST *srcreg, address*
- Arithmetic and Logical Instructions
(mnemonic *srcreg1, srcreg2, dstreg*)
 - ADD SUB
 - AND OR XOR NOT
 - SLL SRL SLE SGT SGE SEQ SNE
 - SLT

7

Instruction Set and Assembling Language (2)

- Mask Stack and Responder Instructions
 - SETMSK
 - TOPMSK
 - POPMSK
 - POPTHEM
 - RPCMSK
 - PUSHMSK
 - PUSHTHEM
 - PUSHMSKTHEM
 - STKTOMEM
 - MEMTOSTK
 - FIND
 - STEP
 - RESFST

8

Instruction Set and Assembling Language (3)

- Maximum and Minimum Searching Instructions

- SETMXMI
- LDMXMI
- STMXMI
- MAX
- MIN

- Branch/Jump Instructions

- BNR
- BRS
- J

9

Associative Operations

Related PE Components:

- **The Responder Register:** to indicate whether a PE is a responder to a particular associative search or not
- **The Step/Find/ResolveFirst Unit:** to support processing multiple responders in various ways
- **The Mask Stack:** to represent at most 16 levels of association. The top of the Mask Stack always represents the current status of the PE – whether it is masked ('1') or unmasked ('0')

10

Example of Associative Search:

Find all Focus cars located in Ohio

- Perform the comparison: model == "Focus", and store the result either '1' or '0' into \$LR1
- Perform the comparison: location == "Ohio", and store the result into \$LR2
- AND \$LR1 with \$LR2, and store the result into the Responder Register

(Note: all the instructions above performed by all PEs in parallel are called unmasked instructions)

11

Unmasked and Masked Instructions

- Unmasked Instruction:
Executed by all the PEs regardless of the state of the Mask Stack
- Masked Instruction:
Executed only by those PEs with a '1' on the top of their Mask Stack

12

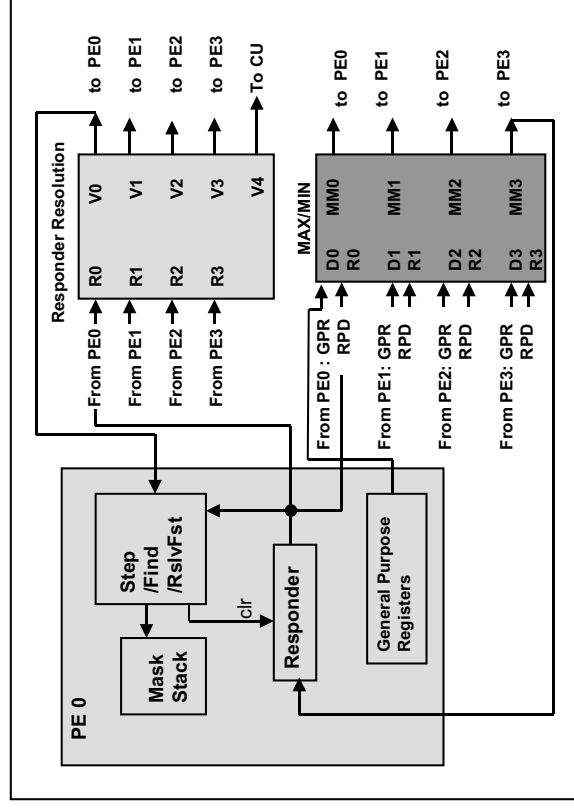
Example of Associative Search Using Masked Instructions:

(Find all Focus cars located in Ohio)

- Initialize the top of the Mask Stack to '1'
- Perform the comparison: `model == "Focus"`, and store the result '1' or '0' into \$LR1
- Perform the comparison: `location == "Ohio"`, and store the result into \$LR2
- AND \$LR1 with \$LR2, and store the result into the Responder Register
- AND the Responder Register with the top of the Mask Stack, and push the ANDing result into the Mask Stack and also store it into the Responder Register
- Increase the rebate of all Focus cars in Ohio by 10 (masked instruction)

13

The MAX/MIN Circuitry, the Responder Resolution Circuitry, and PE₃



14

Using the Falkoff Algorithm for MAX/MIN Search

Maximum-Value Searching (the following steps are performed in parallel for all the data)

- Search bit slices of the data from the most significant bit to the least significant bit:
As each bit slice is processed, each bit is ANDed with a corresponding MM bit (a 1-bit register used to indicate whether or not a data item is the maximum after processing a bit)
- Check the results of the AND to ensure that at least one new maximum value remains:

15

Using the Falkoff Algorithm for MAX/MIN Search (continued)

If this condition is true, then the MM bits are updated by the results of AND; if all the results are 0, then the MM bits are not updated at this time

- Continue to process the remaining bit slices as above until all bits are processed
- After the least significant bit slice is processed: If only one MM bit is '1', it marks the largest number; if more than one MM bit is '1', those data are tied for the maximum value

16

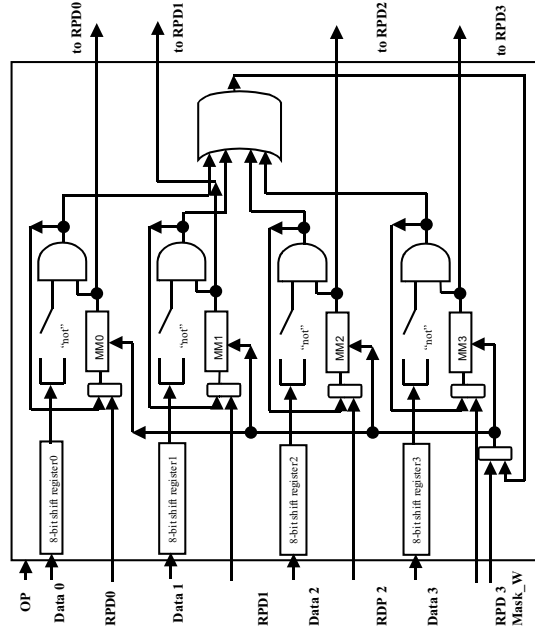
Using the Falkoff Algorithm for MAX/MIN Search (continued)

Minimum-Value Searching:

- Similar to maximum value searching, but complement the bit slices each time before ANDing it with MM bits

17

MAX/MIN Circuit using the Falkoff Algorithm



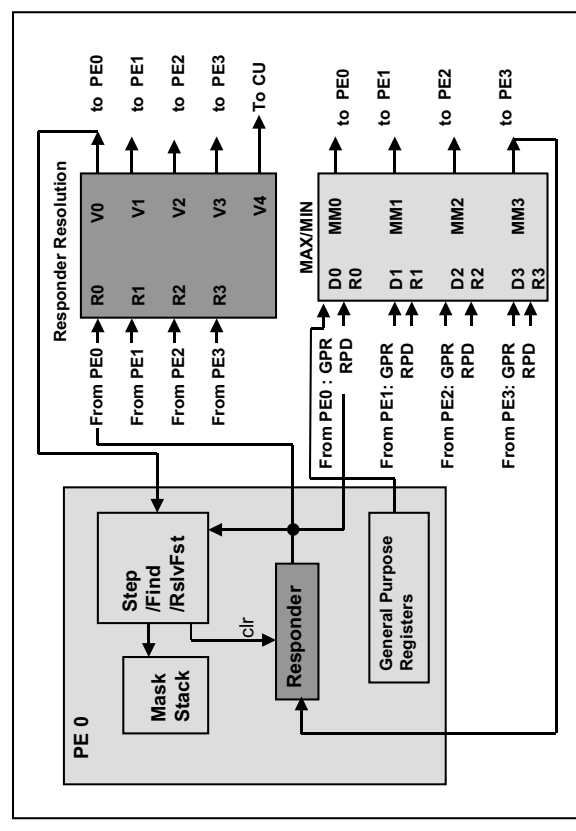
19

Search For the Maximum Rebate in the Data Memories

| Bit Slices (7..0) of Rebates | | Values in MM bits During Processing | |
|------------------------------|----------|-------------------------------------|-----------------------|
| Process bit from MSB to LSB | | After processing each bit | |
| (rebate) | 76543210 | Initialize | 7 6 5 4 3 2 1 0 |
| (170) | 10101010 | (MM0) | 1 1 1 1 0 0 0 0 |
| (160) | 10100000 | (MM1) | 1 1 1 1 0 0 0 0 |
| (190) | 10111110 | (MM2) | 1 1 1 1 1 1 1 1 (max) |
| (180) | 10110100 | (MM3) | 1 1 1 1 1 0 0 0 |

18

The MAX/MIN Circuitry, the Responder Resolution Circuitry, and PE₃



20

Functionality of Responder Resolution Circuit

- Responder resolution:
Send an At-Least-One-Responder signal to the IS control unit
- Support responder selection:
Send a corresponding Responder_Before_Me signal to each PE's Find_Step_ResolveFirst unit

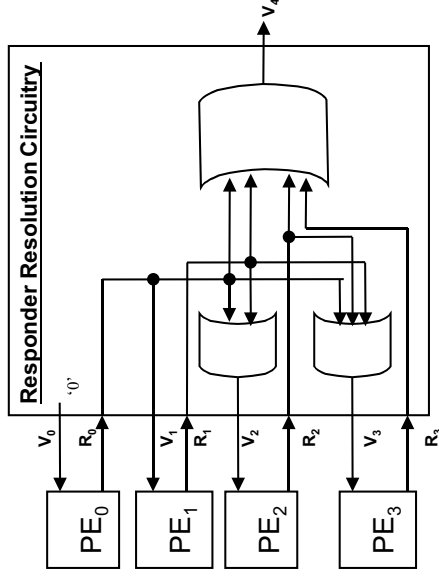
21

Responder Processing

- Process responders in parallel:
 - use masked instructions
- Process responders sequentially:
 - Need some responder selection instructions
 - Need a responder selection mechanism

23

The Responder Resolution Circuitry for 4 PEs



R₀ to R₃ : from responder registers

V₀ to V₃ : called Responder_Before_ME

V₄ : called At_Least_One_Responder

22

Responder Selection Instructions

- Step
repetitively used to pick one responding PE each time for further processing – “for” loop
e.g., to step through all the Focus cars in Ohio to list the features available on each car
- Find
select a responding PE, while still keeping all responders identifiable – “while” loop
e.g., retrieve the tax rate from one of the cars located in OH, then increment the tax rate by a certain amount, afterwards apply this new tax rate to all the cars located in OH

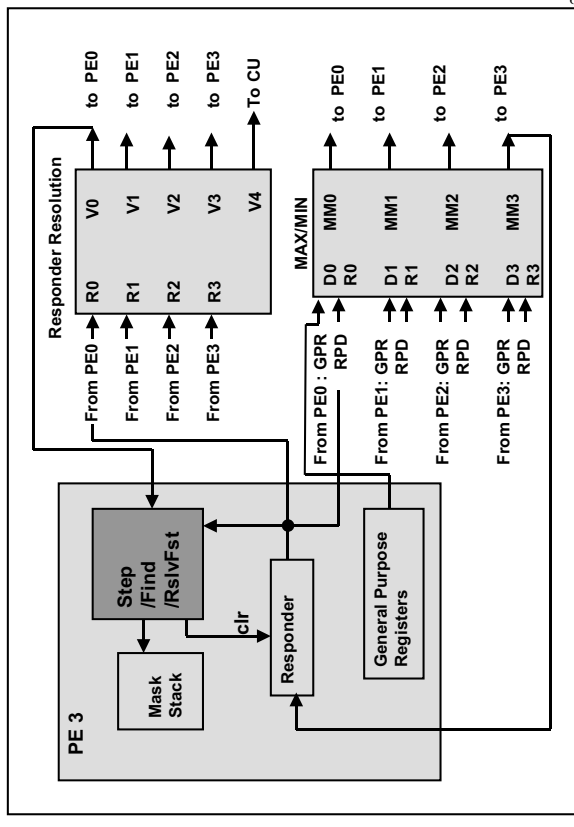
24

Responder Selection Instructions (continued)

- **ResolveFirst**
select a responder and only keep this responder identifiable
e.g., resolve one PE from several PEs which have the values tied for the maximum value

25

The Responder Resolution Circuitry, MAX/MIN Circuitry, and PE₃



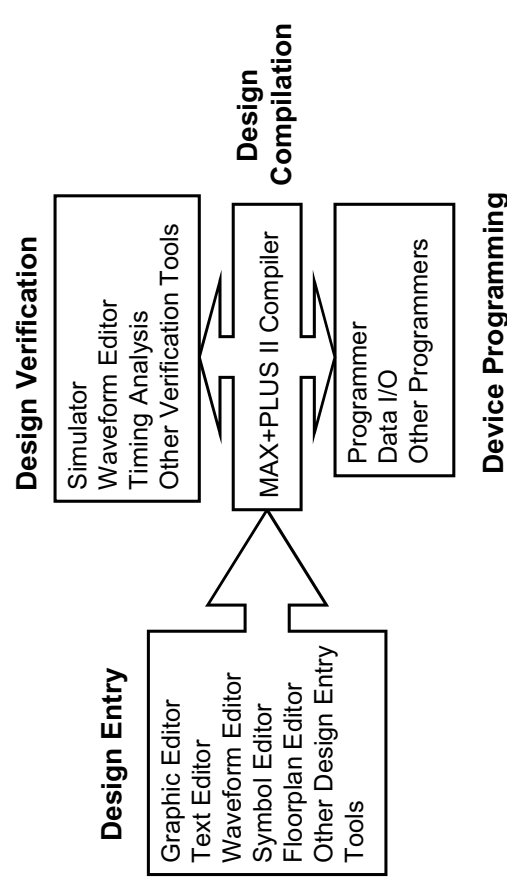
26

Design Language: VHDL

- A standard hardware description language used to model and design digital hardware
 - Support concurrent events
 - can be translated into hardware by some design tools
- good for managing large design structures
- Supported by many CAD tool and programmable logic vendors

27

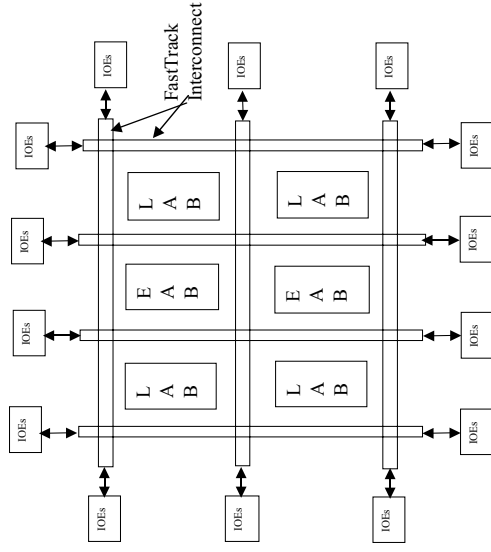
Altera MAX+PLUS II Development System



28

Altera FLEX 10K FPLD

- FLEX10K70 Device:
 - 3,744 LEs
 - 9 EABs
 - 70,000 gates totally



(IOEs – I/O elements)

Partial FLEX10K20 FPLD Architecture

29

Simulation on FLEX 10K 70 Chip

- The ISCU runs at about 10MHz using 50% logical gates
- One EAB is used as a local memory for one PE; 4 PEs and the support circuit runs at about 14MHz using 82% logical cells.

From the simulation result, we can see that the FLEX 10K 70 chip isn't large enough for the 4-PE processor. So our current work is targeting on Altera APEX 20K devices with 1million gates in one chip.

30

Future Work

- Explore more arithmetic features and associative operations
- Develop the complete ASC assembly language and the ASC back-end compiler
- Implement the PE cell interconnection network
- Implement the whole ASC processor on bigger and faster FPGA chips
- Develop the multiple instruction stream MASC model

31