



***Best known methods for
using Cadence Conformal LEC at Intel***

Erik Seligman and Itai Yarom, Intel Corp.

Outline



- Formal Equivalence Verification (FEV) in Intel flows
- Best Known Methods (BKMs) for FEV using Conformal LEC
- Applying to an Intel design
- The future of FEV



Outline



- **Formal Equivalence Verification (FEV) in Intel flows**

- Best Known Methods (BKMs) for FEV using Conformal LEC
- Applying to an Intel design
- The future of FEV

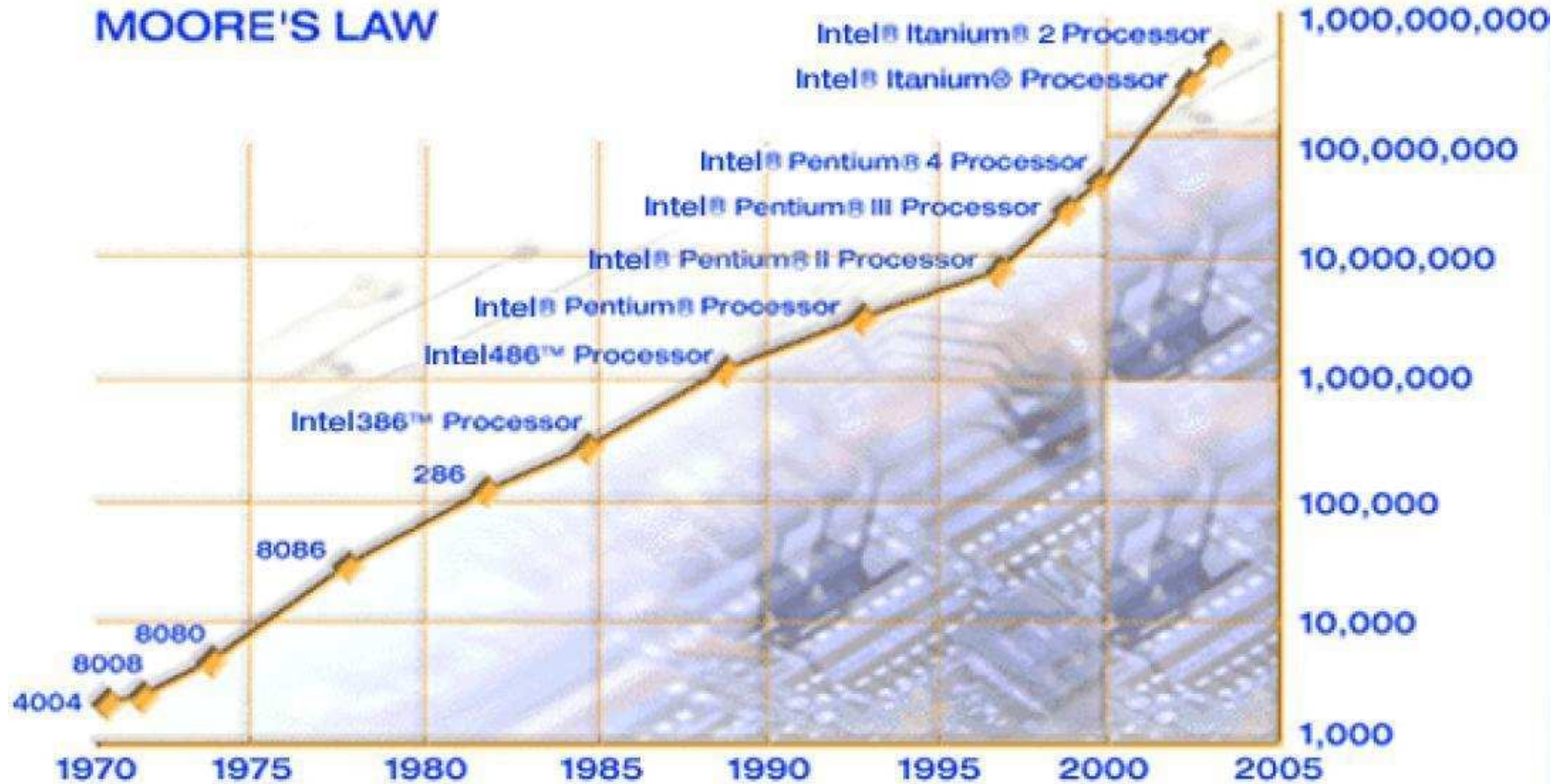


Moore's Law and Us...

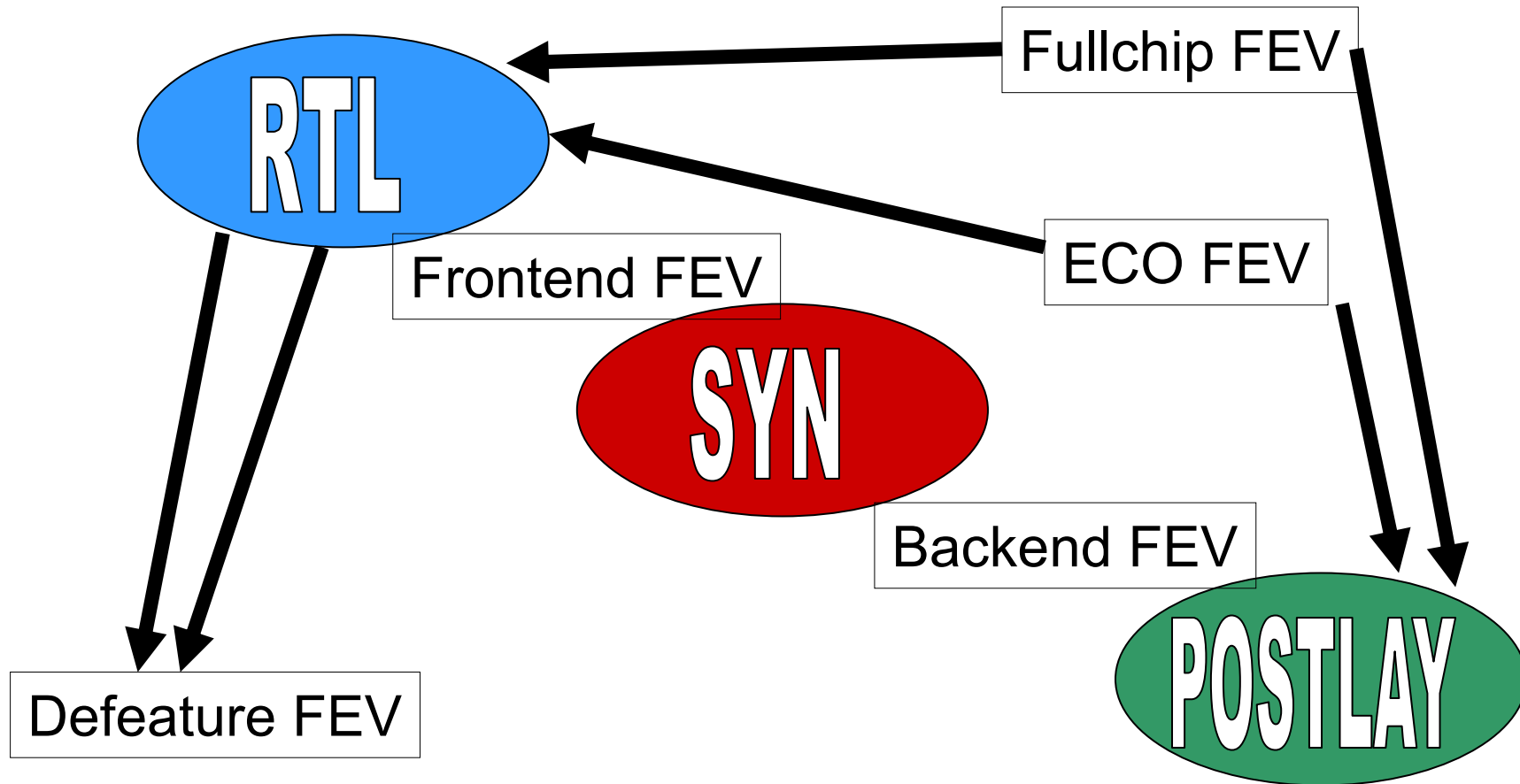


transistors

MOORE'S LAW



Where do we use FEV?



Outline



- Formal Equivalence Verification (FEV) in Intel flows
- **Best Known Methods (BKMs) for FEV using Conformal LEC**
- Applying to an Intel design
- The future of FEV



LEC Challenges

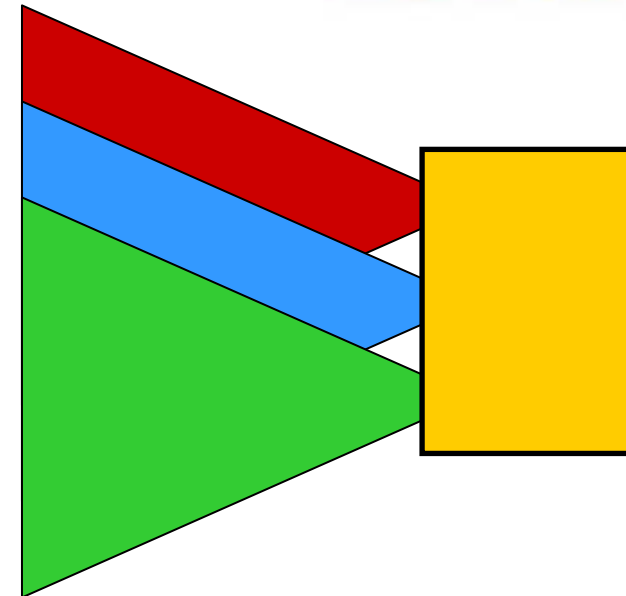


- Abort points
- Non-equivalent points
- Overall confidence for tapeout



Abort Points

- What is an abort point?
 - Abort point is an incomplete comparison point.
- Why do we have abort points?
 - Large models
 - Complex logic optimizations
 - Large don't-care space



BKMs For Handling Abort Points



- Divide and Conquer: *write hier_compare dofile*
 - Effective but messy: prefer other techniques
- Ask LEC to solve the problem for you
 - *set compare effort*: 'secret' levels not in docs!
 - *analyze datapath -merge [-effort high] [-qual 33]*
 - Partitioning / case splitting
- Library cells: Be aware of representation
- Know your don't-care space
- Parallel compare



Non-Equivalent Points, The Never-Ending Battle



- Be careful about obscure or unsupported Verilog
 - $\{a, \{2-\text{WIDTH}\}1'b0, b\} == \{a,b\}$ or $\{a,0,b\}$?
- Use *set flatten model* with care
 - Flatten different-looking flops to identical latches?
 - Lack of *seq_constant* causing unrealistic failures?
- Know your don't-care space
 - May need explicit *\$constraint* for unused opcodes



False Positives: What Keeps FEV Owners Awake at Night



- Need careful audits of FEV runs, don't trust DEs!
- Are all constraints & ignores valid?
 - \$constraint, *constraint*, *stuck_at*, *ignore*...
 - What about contradictions?
- Are all Unreachables understood?
 - Backend team not happy with missing bonus/decap
- Are all bboxes and library cells accounted for?
 - Ownership confusion can result in FEV holes



Outline



- Formal Equivalence Verification (FEV) in Intel flows
- Best Known Methods (BKMs) for FEV using Conformal LEC
- **Applying to an Intel design**
- The future of FEV



Removing the Abort Points



- Initial result: 4 Abort points
 - *analyze abort* gave some hints
- Lec -ultra
 - Enables you to handle a wide variety of datapath structures required for high-performance designs
- analyze datapath:
 - We used -qual 30 to fit all the abort points with success rate over 30% to use this algorithm.
 - Available from version **5.1** when using the Ultra flag. Cadence is working to automate this process.

```
// Command: analyze datapath -merge -verbose -share -qual 30
// Note: add_683(clustered): quality evaluated 45% success
// Note: add_669(clustered): quality evaluated 33% success
// Note: mult_166: quality evaluated 100% success
```



Results



- set compare effort complete
 - The compare effort can be one of the following levels: low, med, high, super, ultra, complete
- compare -single
 - Doesn't share comparison results of share logic.
- Results:

```
• =====  
• Compared points      PO      DFF      DLAT      BBOX      Total  
• -----  
• Equivalent           177      36314    13        189        36693  
• =====  
• CPU time           : 45034.57 seconds (12.51 Hours)  
• Memory usage       : 795.77 M bytes
```



Outline



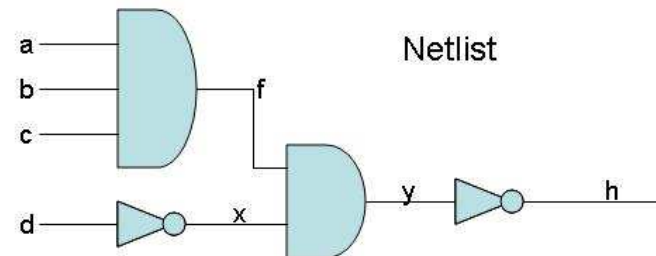
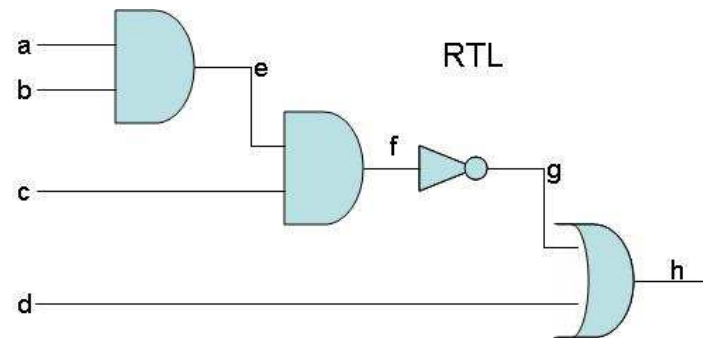
- Formal Equivalence Verification (FEV) in Intel flows
- Best Known Methods (BKMs) for FEV using Conformal LEC
- Applying to an Intel design
- ***The future of FEV***



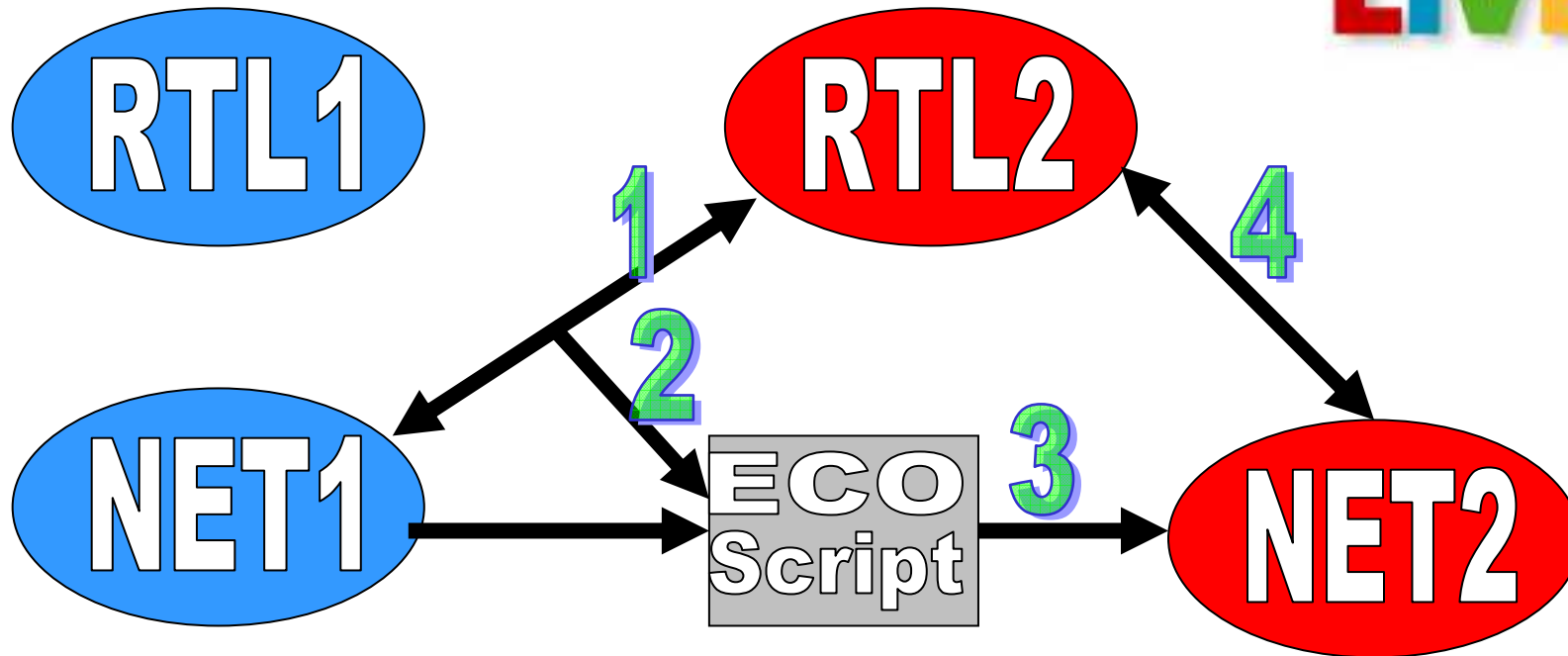
The ECO Problem



- Late change: RTL and Netlist are already here
 - Don't want to resynthesize
- Manually change RTL
 - Also manually change netlist?
- Logic not preserved
 - Signals 'e' and 'g' are gone!
 - What if ECO adds NOT at e?
 - Manual netlist edit hard



ECO Automation Flow



1. Use LEC to identify the areas that were changed.
2. Generate a list of those areas and convert to an ECO script.
3. Replace the old logic with the ECO logic (using ECO script).
4. Verify that the new netlist is logically equivalent to the RTL w/ ECO using LEC.



Sequential FEV



- Industry moving towards high-level modeling
 - Major FEV gap: few solutions for HLM vs RTL/Netlist
 - Relying on simulation very dangerous, but common
 - Sequential FEV: relax state-matching constraint
 - LEC-DP 'analyze retiming' is a baby step
 - Competitors starting to develop real solutions
- Sequential formal engines are here
 - Formal Property Verification (FPV) tools common
 - Cadence Incisive, for example
 - FEV + FPV: converging?



Conclusions



- FEV is critical to Intel design flows
- Conformal LEC is a very useful tool for ASIC FEV
- FEV process is complex
 - Share BKM's: Aborts, Non-Eqs, design closure
 - Initially infeasible designs can often be handled
- Many directions for FEV to continue to develop
 - ECO Automation, Sequential Equivalence
 - If Conformal doesn't address them, other tools will!





Thank You!

***Best known methods for
using Cadence Conformal LEC at Intel***

Erik Seligman and Itai Yarom, Intel Corp.

Increasing the compare effort



- LEC is tuned to address common comparison logic.
 - This is efficient for most runs.
 - To address abort points you want to increase the comparison effort:

```
analyze datapath -merge -verbose -share
```

```
analyze abort -compare
```

```
set compare effort complete
```

```
compare -single
```



Initial Comparison Results

With version 05.20-s220 (07-Mar-2006)



```

. =====
. Compared points      PO      DFF      DLAT      BBOX      Total
. -----
. Equivalent          177      36314* 13        185        36689
. -----
. Abort                0        0        0         4         4
. =====
. The busses din[67..0] & dft_din[67..0] are the inputs of each of the aborted points

```

```

> > analyze abort -compare
> =====
> Analysis Results
> =====
> Compared result is abort.
> (G) + 37034 BBOX /u_common/.../u_mem
> (R) + 37535 BBOX /u_common/.../u_mem
> (G) + 331158 BUF /u_common/.../u_mem/din[67]
> (R) + 456191 BUF /u_common/.../u_mem/din[67]
> =====
> Region      Size      Support      ADD/SUB      MUL/DIV      DC      Corr
>              gates, mod  gates, mod
> -----
> Golden      30751      7161        304, 3       0, 0         150      2848
> Revised     39672      7163         0, 0         0, 0         0        2945
> Proved      14509      7161        14, 1        0, 0         0         -
> =====

```



Abort Point Details



```

> =====
> Region      Size      Support      ADD/SUB      MUL/DIV      DC      Corr
>                                     gates, mod   gates, mod
> -----
> Golden      30751      7161         304, 3       0, 0         150      2848
> Revised     39672      7163         0, 0         0, 0         0         2945
> Proved      14509      7161         14, 1        0, 0         0         -
> =====
  
```

> Legends:

- > **Proved region:** region proved to be equivalent.
- > **Golden region:** region in golden design that is outside the proved region.
- > **Revised region:** region in revised design that is outside the proved region.
- > **Size:** the number of gates in the region.
- > **Support:** the number of support points (logic cone fan-in).
- > **ADD/SUB:** the number of gates and modules from adders and/or subtractors.
- > **MUL/DIV:** the number of gates and modules from multipliers and/or dividers.
- > **DC:** the number of don't care gates.
- > **Corr:** the number of gates with possible correspondence gates.



The Design

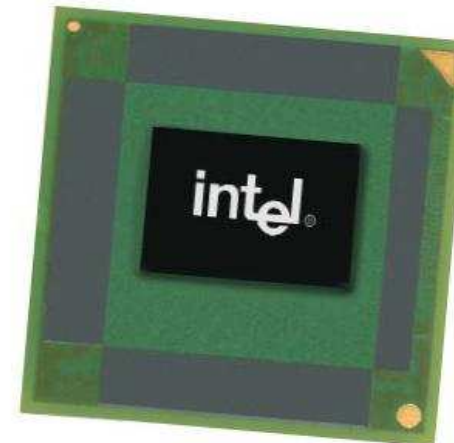


Intel® Gigabit Ethernet Controller



High-performance, Dual-Port Gigabit Network Connectivity for Servers and Embedded System Designs

- High-performing, PCI Express* 10/100/1000 Ethernet connection
- Dual-port, single-chip configuration simplifies designs
- Footprint compatibility with single-port Gigabit Ethernet (GbE) controllers for flexible designs





Introducing Intel® vPro™ Technology
A leap forward in business PCs.



Thursday, September
21, 2006

Best known methods for using
Cadence Conformal LEC at Intel

26

Intel vPro

Manageability

- Asset Inventory: Finding systems remotely
- Remote management for problem resolution

Security

- Three layers of defense
- Filtering threats and isolating PCs
- Push updates and patches down the wire regardless of PC power state

Energy efficient performance

- Intel® Core™ microarchitecture
- Cutting-edge transistor technologies
- Energy-efficient technologies

Feature

Benefit

Out-of-band (OOB) system access

Allows remote management of PCs regardless of system power** or OS state

Remote troubleshooting and recovery

Significantly reduces deskside visits to increase the efficiency of IT technical staff

Proactive alerting

Accelerates problem detection and decreases end-user downtime

Remote HW and SW asset tracking

Increases speed and accuracy over manual inventory tracking, reducing asset accounting costs

Third-party nonvolatile storage

Eliminates reliance on local software agents to store and retrieve data to help prevent accidental data loss

Proactive blocking and reactive containment of network threats

Helps prevent certain viruses and worms from infecting end-user PCs and spreading, increasing network uptime

** Requires power supply and an active network connection.



Intel vPro Technology.

A Leap Forward in Business PCs.

> MANAGEABILITY

> SECURITY

> ENERGY-EFFICIENT PERFORMANCE

