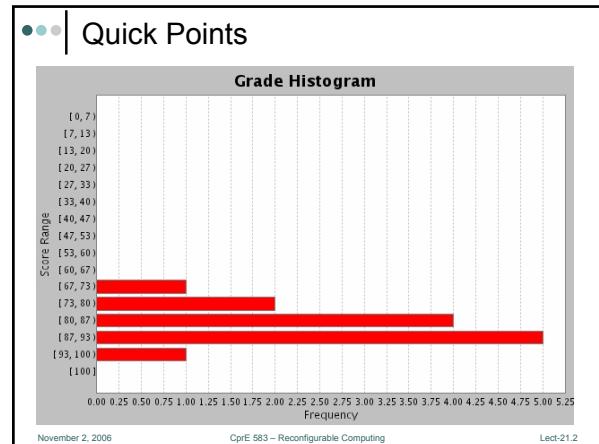


CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

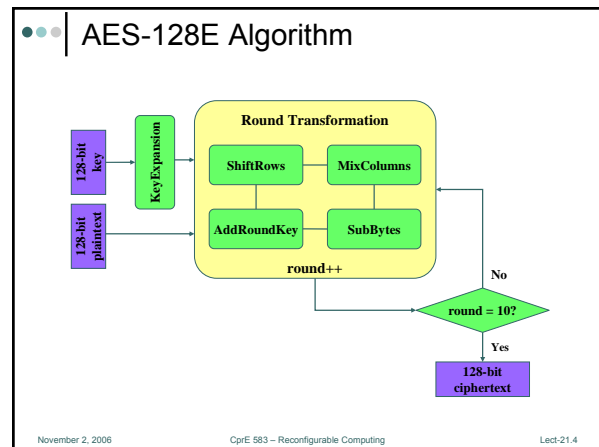
Lecture #21 – HW/SW Codesign



HW #4 Discussion

- Problem 1 – did just a simple adder work?
- Problem 2 – how did you implement the permutation table?
- Problem 3 – did you use a counter?

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.3



Overview of AES (cont.)

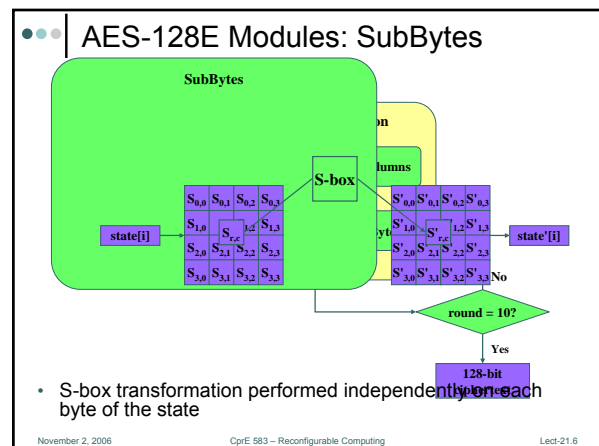
- 128-bit input is copied into a two-dimensional (4x4) byte array referred to as the *state*
 - Round transformations operate on the state array
 - Final state copied back into 128-bit output
- AES makes use of a non-linear substitution function that operates on a single byte
 - Can be simplified as a look-up table (*S-box*)

```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303
304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319
320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335
336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351
352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367
368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383
384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399
400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415
416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447
448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463
464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479
480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511
512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527
528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543
544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559
560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575
576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591
592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607
608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623
624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639
640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655
656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671
672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687
688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703
704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719
720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735
736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751
752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767
768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783
784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799
800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815
816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831
832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847
848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863
864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879
880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895
896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911
912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927
928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943
944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959
960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975
976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991
992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006
1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048
1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062
1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076
1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090
1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104
1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118
1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132
1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146
1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160
1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174
1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188
1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202
1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216
1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230
1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244
1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258
1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272
1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286
1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300
1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314
1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342
1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356
1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370
1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384
1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398
1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412
1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426
1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454
1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468
1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482
1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496
1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510
1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524
1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538
1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552
1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566
1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580
1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594
1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608
1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622
1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636
1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650
1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664
1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678
1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692
1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706
1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720
1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734
1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748
1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762
1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776
1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790
1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804
1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818
1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832
1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846
1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860
1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874
1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888
1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902
1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916
1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930
1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944
1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958
1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972
1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986
1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000

```

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.5



••• AES-128E Modules: ShiftRows

Bytes in the last three rows of the state are shifted cyclically over variable offsets

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.7

••• AES-128E Modules: MixColumns

Modulo polynomial-basis multiplication performed on each column of the state

Can be simplified as series of AND and XOR operations

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.8

••• MixColumns Implementation

```

entity MixColumns is
    port (STATE_IN : in STATEtype;
          RNUM_IN : in RNUMtype;
          STATE_OUT : out STATEtype);
end MixColumns;

architecture behavior of MixColumns is
    signal tSTATE : STATEtype;
begin
    process (STATE_IN)
        variable t1, t2 : std_logic_vector(7 downto 0);
    begin
        for i in 0 to 3 loop
            for j in 0 to Nb-1 loop
                -- Multiply by 2
                t1 := STATE_IN(i mod 4)(j)(6 downto 0) & '0';
                if (STATE_IN(i mod 4)(j)(7) = '1') then
                    t1 := t1 xor x"1b";
                end if;

                -- Multiply by 3
                t2 := STATE_IN((i+1) mod 4)(j)(6 downto 0) & '0';
                if (STATE_IN((i+1) mod 4)(j)(7) = '1') then
                    t2 := t2 xor x"1b";
                end if;
                t2 := t2 xor STATE_IN((i+1) mod 4)(j);
                tSTATE(i)(j) <= t1 xor t2 xor STATE_IN((i+2) mod 4)(j) xor STATE_IN((i+3) mod 4)(j);
            end loop;
        end loop;
    end process;
end behavior;

```

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.9

••• AES-128E Modules: AddRoundKey

Words from the round-specific key are XORed into columns of the state

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.10

••• AddRoundKey Implementation

```

entity AddRoundKey is
    port (STATE_IN : in STATEtype;
          KEY_IN : in KEYtype;
          STATE_OUT : out STATEtype);
end AddRoundKey;

architecture behavior of AddRoundKey is
begin
    process (STATE_IN, KEY_IN)
    begin
        for j in 0 to (Nb-1) loop
            STATE_OUT(0)(j) <= STATE_IN(0)(j) xor KEY_IN(j)(31 downto 24);
            STATE_OUT(1)(j) <= STATE_IN(1)(j) xor KEY_IN(j)(23 downto 16);
            STATE_OUT(2)(j) <= STATE_IN(2)(j) xor KEY_IN(j)(15 downto 8);
            STATE_OUT(3)(j) <= STATE_IN(3)(j) xor KEY_IN(j)(7 downto 0);
        end loop;
    end process;
end behavior;

```

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.11

••• AES-128E Modules: KeyExpansion

Initial 128-bit key is converted into separate keys for each of the 10 required rounds

Consists of Sbox transformations and some XORs

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.12

Design Decisions

- Online/offline key generation
- Inter-round layout decisions
 - Round unrolling
 - Round pipelining
- Intra-round layout decisions
 - Transformation pipelining
 - Transformation partitioning
- Technology mapping decisions
 - S-box synthesis as Block SelectRAM, distributed ROM primitives, or logic gates

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.13

Round Unrolling / Pipelining

- *Unrolling* replaces a loop body (round) with N copies of that loop body
- AES-128E algorithm is a loop that iterates 10 times – $N \in [1, 10]$
 - $N = 1$ corresponds to original looping case
 - $N = 10$ is a fully unrolled implementation
- *Pipelining* is a technique that increases the number of blocks of data that can be processed concurrently
 - Pipelining in hardware can be implemented by inserting registers
 - Unrolled rounds can be split into a certain number of pipeline stages
- These transformations will increase throughput but increase area and latency

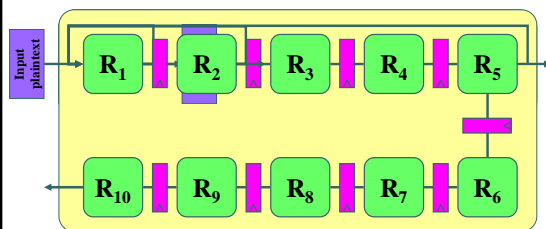
November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.14

Round Unrolling / Pipelining (cont.)

Unrolling factor = 10
Round pipelining = ON



November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.15

Transformation Partitioning/Pipelining

- FPGA maximum clock frequency depends on critical logic path
 - Inter-round transformations can't improve critical path
 - Individual transformations can be pipelined with registers similar to the rounds
- Transformations that are part of the maximum delay path can be partitioned and pipelined as well
- Can result in large gains in throughput with only minimal area increases

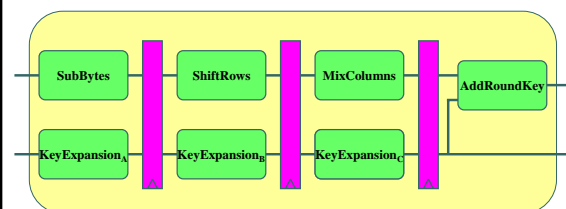
November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.16

Partitioning / Pipelining (cont.)

Transformation pipelining = ON
Transformation partitioning = ON



November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.17

S-box Technology Mapping

- With synthesis primitives, can map the S-box lookup tables to different hardware components
- Two S-boxes can fit on a single Block SelectRAM

```
constant Ssynromstyle: string := "select_rom"; -- {logic, select_rom}
entity Sbox is
    port(BYTE_IN : in std_logic_vector(7 downto 0);
         BYTE_OUT : out std_logic_vector(7 downto 0));

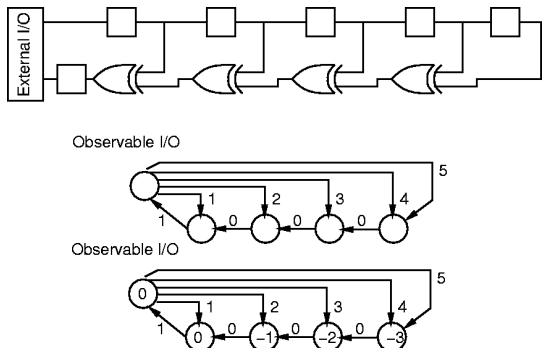
    attribute syn_romstyle : string;
    attribute syn_romstyle of BYTE_OUT : signal is Ssynromstyle;
end Sbox;
-- Sample VHDL code
...
```

November 2, 2006

CprE 583 – Reconfigurable Computing

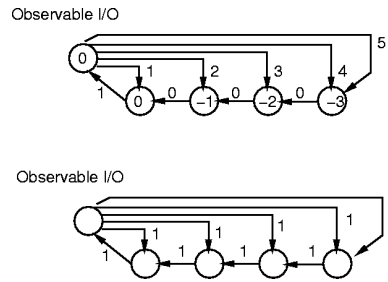
Lect-21.18

Recap – Retiming



November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.19

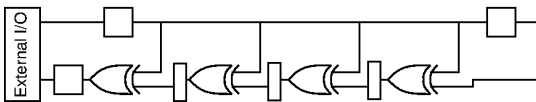
Recap – Retiming (cont.)



$$\text{weight}(e') = \text{weight}(e) + \text{lag}(\text{head}(e)) - \text{lag}(\text{tail}(e))$$

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.20

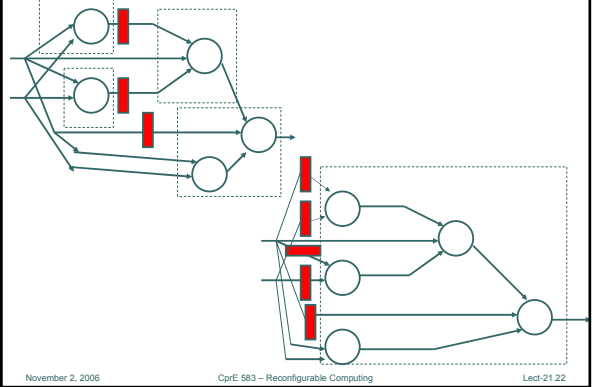
Retiming and Pipelining



- Can use this retiming to pipeline
- Assume have enough (infinite supply) of registers at edge of circuit
- Retime them into circuit
- See [WeaMar03A] for details

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.21

Recap – Retiming and Covering



November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.22

Outline

- HW #4 Discussion
- Recap
- HW/SW Codesign
 - Motivation
 - Specification
 - Partitioning
 - Automation

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.23

Hardware/Software Codesign

- Definition 1 – the concurrent and co-operative design of hardware and software components of an embedded system
- Definition 2 – A design methodology supporting the cooperative and concurrent development of hardware and software (co-specification, co-development, and co-verification) in order to achieve shared functionality and performance goals for a combined system [MicGup97A]

November 2, 2006 CprE 583 – Reconfigurable Computing Lect-21.24

Motivation

- Not possible to put everything in hardware due to limited resources
- Some code more appropriate for sequential implementation
- Desirable to allow for parallelization, serialization
- Possible to modify existing compilers to perform the task

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.25

Why put CPUs on FPGAs?

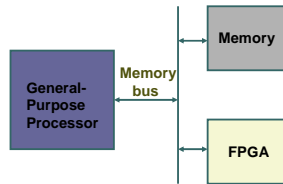
- Shrink a board to a chip
- What CPUs do best:
 - Irregular code
 - Code that takes advantage of a highly optimized datapath
- What FPGAs do best:
 - Data-oriented computations
 - Computations with local control

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.26

Computational Model



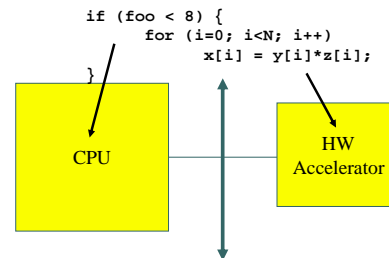
- Most recent work addressing this problem assumes relatively slow bus interface
- FPGA has direct interface to memory in this model

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.27

Hardware/Software Partitioning



November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.28

Methodology

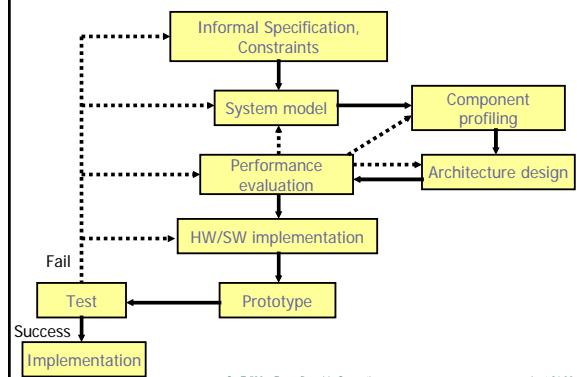
- Separation between function, and communication
- Unified refinable formal specification model
 - Facilitates system specification
 - Implementation independent
 - Eases HW/SW trade-off evaluation and partitioning
- From a more practical perspective:
 - Measure the application
 - Identify what to put onto the accelerator
 - Build interfaces

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.29

System-Level Methodology



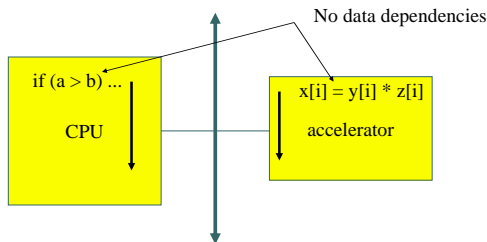
November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.30

Concurrency

- Concurrent applications provide the most speedup



November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.31

Partitioning

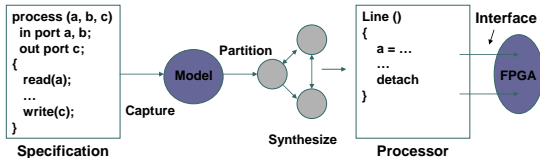
- Can divide the application into several processes that run concurrently
- Process partitioning exposes opportunities for parallelism

```

if (i>b) ... Process 1
for (i=0; i<N; i++) ... Process 2
for (j=0; j<N; j++) ... Process 3
    
```

November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.32

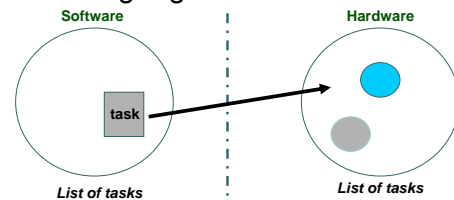
Automating System Partitioning



- Good partitioning mechanism:
 - Minimize communication across bus
 - Allows parallelism → both hardware (FPGA) and processor operating concurrently
 - Near peak processor utilization at all times (performing useful work)

November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.33

Partitioning Algorithms

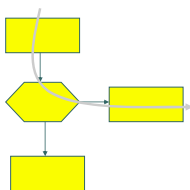


- Assume everything initially in software
- Select task for swapping
- Migrate to hardware and evaluate cost
 - Timing, hardware resources, program and data storage, synchronization overhead
- Cost evaluation and move evaluation similar to what we've seen regarding mincut and simulated annealing

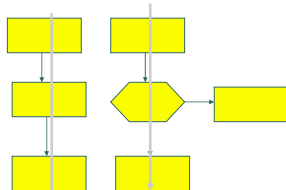
November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.34

Multi-threaded Systems

- Single thread:



- Multi-thread:



November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.35

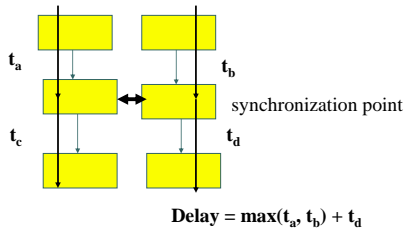
Performance Analysis

- Single threaded:
 - Find longest possible execution path
- Multi-threaded with no synchronization:
 - Find the longest of several execution paths
- Multi-threaded with synchronization:
 - Find the worst-case synchronization conditions

November 2, 2006 CprE 583 - Reconfigurable Computing Lect-21.36

Multi-threaded Performance Analysis

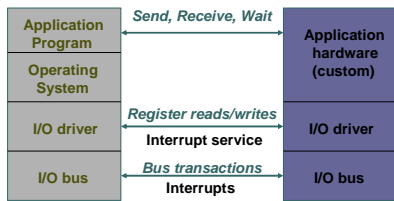
- Synchronization causes the delay along one path to affect the delay along another



Control

- Need to signal between CPU and accelerator
 - Data ready
 - Complete
- Implementations:
 - Shared memory
 - Handshake
- If computation time is very predictable, a simpler communication scheme may be possible

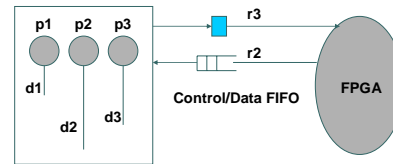
Communication Levels



- Easier to program at application level
 - (send, receive, wait) but difficult to predict
- More difficult to specify at low level
 - Difficult to extract from program but timing and resources easier to predict

Other Interface Models

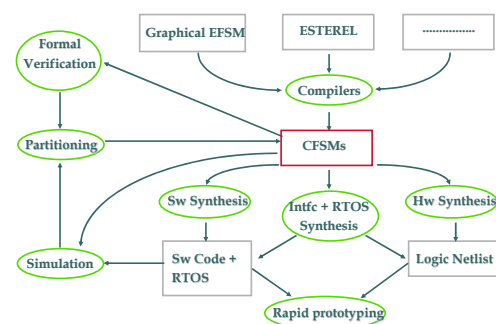
- Synchronization through a FIFO
- FIFO can be implemented either in hardware or in software
- Effectively reconfigure hardware (FPGA) to allocate buffer space as needed
- Interrupts used for software version of FIFO



Debugging

- Hard to test a CPU/accelerator system:
 - Hard to control and observe the accelerator without the CPU
 - Software on CPU may have bugs
- Build separate test benches for CPU code, accelerator
- Test integrated system after components have been tested

POLIS Codesign Methodology



••• | Codesign Finite State Machines

- POLIS uses an FSM model for
 - Uncommitted
 - Synthesizable
 - Verifiable
 Control-dominated HW/SW specification
- Translators from
 - State diagrams,
 - Esterel, ECL, ReactiveJava
 - HDLs
 Into a single FSM-based language

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.43

••• | CFSM behavior

- Four-phase cycle:
 - ☆ Idle
 - ⌚ Detect input events
 - ⌚ Execute one transition
 - ⌚ Emit output events
- Software response could take a long time:
 - Unbounded delay assumption
- Need efficient hw/sw communication primitive:
 - Event-based point-to-point communication

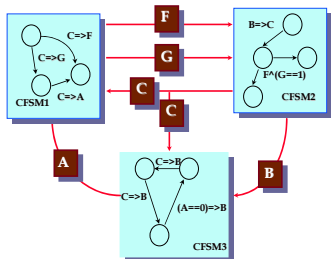
November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.44

••• | Network of CFSMs

- Globally Asynchronous, Locally Synchronous (GALS) model



November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.45

••• | Summary

- Hardware/software codesign complicated and limited by performance estimates
- Algorithms not generally as good as human partitioning
- Other interesting issues include dual processors, special memory interfaces
- Will likely evolve at faster rate as compilers evolve

November 2, 2006

CprE 583 – Reconfigurable Computing

Lect-21.46