CLA (1) - 다음으로 4비트 carry look ahead adder에 대해 설명하겠습니다. 앞서 설명해드린 4비트 병렬 가산기를 구성하는 Full Adder는 여러 게이트들로 구성되어 있고 이 각각의 게이트들은 gate delay가 존재합니다. 따라서 carry값을 구하기 위해 이전의 carry값들의 연산 과정을 모두 기다려야 하는 4bit 병렬 가산기는 연산과정에서 gate delay가 많이 발생하게 되어 시간이 오래 걸린다는 단점이 있습니다.

CLA(2) - 이러한 병렬 가산기의 속도 문제를 해결하기 위한 방법이 CLA라고 하는 Carry Look Ahead Adder입니다. CLA Adder는 병렬 가산기에서 시간 측면의 단점을 발생시킨 carry 연산 부분을 그림에서처럼 CLA generator로 대체하여 속도의 문제를 해결합니다.

CLA(3) - CLA generator의 연산을 위해서 두 가지의 함수가 필요한데 하나는 carry generate 함수로 논리식은 gi=xiyi이고 다른 하나는 carry propogate 함수로 논리식은 pi = xi^yi입니다. 여기서 G함수는 캐리 생성 함수로 기존 연산과 관계 없이 반드시 캐리가 생성됨을 확인하는 함수이고 P함수는 추가적으로 캐리가 발생할 가능성을 검사하는 함수입니다. 이 두 함수로 아래 두 줄을 보시면 Full Adder에서의 두 논리식을 G와 P를 이용해 다음과 같이 새로운 식으로 표현할 수 있습니다.

CLA(4) - 이때 C를 구해주는 식을 보면 좌변에는 Ci+1이 있고 우변에는 Ci가 있는 것을 볼 수 있습니다. 이는 초기 입력으로 주어지는 Ci를 통해 아래 4개와 같이 재귀적으로 Ci+1부터 Ci+4까지 한번에 구할 수 있음을 의미합니다. 따라서 기존의 병렬 가산기처럼 이전의 carry값들의 연산과정을 모두 기다릴 필요 없이 초기 입력값만으로 모든 carry값을 구할 수 있게 되고 이로 인해 carry 연산으로 발생하는 속도 문제를 해결할 수 있게 됩니다.

BCD(1) - 다음으로 BCD adder에 대해 설명하겠습니다. BCD adder는 BCD 코드로 표현된 값들을 더하거나 빼는 연산을 해서 다시 BCD 코드로 표현하는 가산기를 말합니다. BCD코드는 이전 실습에서도 몇 번 다뤘었던 내용이라 모두 아시다시피 십진수 0~9까지를 나타내기 위한 코드이기 때문에 10이상의 십진수는 4비트 이진코드로 표현할 수 없습니다. 따라서 나와있는 왼쪽 예시와 같이 BCD연산에서 피연산자와 연산 결과가 모두 0~9안에 포함되어 있는 값이면 4비트로 표현하는데에 문제가 없지만 오른쪽 예시와 같이 연산 결과가 4비트 이진코드로 표현되지 않을 수도 있습니다.

BCD(2) - 이러한 경우 연산결과에 6에 해당하는 이진코드인 0110을 더해주어 12라는 값을 0001 0010의 BCD 코드로 표현할 수 있게 된다. 여기서 6만큼, 즉 0110만큼 더해주는 것은 BCD 코드에서 표현되는 4비트 이진코드 외에 사용되지 않는 10~15에 해당하는 6개의 4비트 이진코드가 있기 때문에 6만큼을 더해서 그에 맞는 BCD 코드로 표현해주는 것입니다.

BCD(3) - 이렇게 10이상의 두자리 십진수말고도 세자리 십진수도 다음과 같이 BCD 코드로 표현할 수 있습니다.

BCD(4) - 이 그림은 저희가 이번 실습에서 구현하는 BCD adder의 회로도를 나타낸 것입니다. 앞에서 설명해드린 BCD 연산과정을 이 회로도를 통해 다시 확인할 수 있습니다. 먼저 빨간색 네모칸의 부분은 입력으로 들어오는 2개의 4비트 이진수를 Full Adder를 통해 더하는 과정입니다. 여기서 덧셈 결과가 16이상의 값이 나오게 되면 Full Adder에서 carry out에 해당하는 K가 1의 값을 갖게 됩니다.

BCD(5) - 덧셈 후 연산 결과를 s3s2s1s0라고 할 때 연산 결과가 옆의 표에 나와있듯이 10~15에 해당하는 값이면 s3와 s2가 1이거나 s3와 s1이 1이거나 또는 s3,s2,s1이 모두 1의 값을 갖게 됩니다. 따라서 이를 확인하기 위해 s3와 s1, 그리고 s3와 s2를 and 게이트의 input으로 하여 연산결과가 10~15사이의 값인지 확인합니다. 따라서 연산 결과가 16이상이거나 10~15의 값을 갖게 되면 제일 왼쪽의 OR게이트는 1의 값을 출력하게 되고 연산결과가 0~9의 값이면 0의 값을 출력합니다.

BCD(6) - 이렇게 나온 OR게이트의 출력값이 0이면 연산 결과가 그대로 나오게 되고 1이 되면 연산결과인 s3s2s1s0와 그림에 나와있듯이 0110, 앞서 설명드렸들이 6만큼을 더해 연산 결과를 BCD 코드로 표현하게 된다.